

Vysoká škola báňská – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky



**Využití zvukové karty počítače jako nízkofrekvenčního
generátoru**

The Use of a Sound Card as a Low Frequency Generator

2018

Bc. Zbyněk Fojtík

Zadání diplomové práce

Student:

Bc. Zbyněk Fojtík

Studijní program:

N2647 Informační a komunikační technologie

Studijní obor:

2601T013 Telekomunikační technika

Téma:

Využití zvukové karty jako nízkofrekvenčního generátoru
The Use of a Sound Card as a Low Frequency Generator

Jazyk vypracování:

čeština

Zásady pro vypracování:

Zvuková karta počítače poskytuje možnost realizovat levný měřicí přístroj. Cílem práce je navrhnout a realizovat generátor a prověřit jeho vlastnosti.

1. Zpracujte přehled možných způsobů využití PC k realizaci nízkofrekvenčního generátoru. Zaměřte se více na možnost využití zvukové karty pro tento účel.
2. Navrhněte programovatelný generátor časově závislých průběhů s možností nastavit typ průběhu, amplitudu, kmitočet. Výstup je dvoukanálový s nezávislým nastavením každého kanálu.
3. Vlastnosti realizovaného generátoru ověřte laboratorním měřením a srovnajte s vlastnostmi komerčního generátoru užívaného ve školní laboratoři.

Seznam doporučené odborné literatury:


[1] Ďurďa, M. *Využití PC jako osciloskopu*. Ostrava, 2009. Diplomová práce. VŠB-Technická Univerzita Ostrava.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Dr. Ing. Libor Gajdošík**

Datum zadání: 01.09.2016

Datum odevzdání: 30.04.2018


doc. Ing. Miroslav Vozňák, Ph.D.
vedoucí katedry




prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne: *30. dubna 2018*

A handwritten signature in blue ink, appearing to read 'Fostek', written over a dotted line.

podpis studenta

Poděkování

Rád bych poděkoval Dr. Ing. Liboru Gajdošíkovi za odbornou pomoc a konzultaci při vytváření této diplomové práce.

Abstrakt

Cílem práce je navrhnout a realizovat programovatelný generátor a prověřit jeho vlastnosti. V první části této diplomové práce se zabývám možným využitím zvukové karty jako nízkofrekvenčního generátoru.

V této části také porovnávám dvě různé možnosti realizace generátoru nízkých frekvencí pomocí dvou knihoven NAudio a CScore. V druhé části diplomové práce se zabývám návrhem a realizací generátoru s časově závislým průběhem, možností nastavení typu průběhu, amplitudy a kmitočtu. Třetí část ověřuji vlastnosti realizovaného generátoru pomocí měření a srovnávám s vlastnostmi komerčního generátoru. Taktéž porovnávám výsledky na několika různých třídách zvukových karet. Měření bylo provedeno, na přístrojích Tektronix TDS 2012B a Spektrálního analyzáru DYNON INSTRUMENTS ELAB-080, který byl využit také jako generátor signálu pro srovnání s realizovanou softwarovou verzí. V poslední části diplomové práce popisují ovládání navrženého a realizovaného nízkofrekvenčního generátoru SIGEN.

Klíčová slova

Nízkofrekvenční generátor; zvuková karta; osciloskop; vzorkovací teorém; převodník; digitální; analogový; signál; tick; napětí; rozhraní; synchronizace; Fourierova transformace; spektrum; amplituda; frekvence

Abstract

The objective of this thesis is to design and realize a programmable generator and check its properties. In its first part this thesis deals with a possible usage of a sound card as a low-frequency generator.

In this part I also compare two different possibilities of the realization of the low-frequency generator using the two libraries of NAudio and CSCore. The second part of my thesis deals with the design and realization of a generator with a time-dependent course, the possibility of the setting of a course type, amplitude and the frequency of oscillation. The third part verifies the properties of the created generator by means of gauging and I compare the properties with those of a commercial generator. I also compare the results of several different classes of sound cards. The gauging was done on the devices of TDS 2012B and DYNON INSTRUMENTS ELAB-080 Spectral Analyzer, which was also used as a signal generator to compare with a software version. The last part of my thesis describes the operating of the designed and created low-frequency SIGEN generator

Key words

Low Frequency Generator; sound card; oscilloscope; sampling theorem; converter; digital; analog; signal; tick; Tension; interface; synchronization; Fourier transform; spectrum; amplitude; frequency

Seznam použitých symbolů

Symbol	Jednotky	Význam symbolu
U	V	Napětí
T	s	Čas
R	Ω	Odpor
<i>f</i>	Hz	Frekvence
Um	mV	Amplituda
C	F	Farad

Seznam použitých zkratk

Zkratka	Význam
.NET	Je zastřešující název pro soubor technologií v softwarových produktech
C#	Programovací jazyk
MS-PL	Microsoft Public License, sada nástrojů který slouží pro projekty společného vývoje softwaru.
PC	Personal Computer
VoIP	Voice over Internet Protocol
WPF	Windows Presentation Foundation, je knihovna tříd pro tvorbu grafického rozhraní, které je součástí .NET frameworku od verze 3.0
VST	Virtual Studio Technology, instrumenty pro příjem dat

Seznam použitých termínů

Termín	Význam termínu
Elektrický proud	Elektrický proud je uspořádaný pohyb nositelů elektrického náboje, vyjadřuje množství náboje prošlého za jednotku času.
Elektrické napětí	Elektrické napětí mezi dvěma body je definováno jako rozdíl elektrických potenciálů.
Frekvence	Frekvence udává počet opakování periodického děje za daný časový úsek.
Perioda	Perioda udává dobu trvání jednoho opakování periodického děje.
Amplituda	Amplituda je maximální hodnota periodicky měnící se veličiny.

Obsah

Úvod.....	- 12 -
1 Možné využití zvukové karty.....	- 13 -
Volba podporované knihovny pro obsluhu zvukové karty.....	- 14 -
1.1.1 NAudio.....	- 14 -
1.1.2 CSCore.....	- 15 -
1.2 Volba použité knihovny.....	- 15 -
2 Signály.....	- 16 -
2.1 Periodický signál.....	- 16 -
2.1.1 Sinusový signál.....	- 17 -
2.1.2 Hradbový signál.....	- 18 -
2.1.3 Trojúhelníkový signál.....	- 20 -
2.1.4 Pilové signály.....	- 21 -
2.2 Vzorkování signálů.....	- 22 -
2.2.1 USB DAC, aneb externí zvuková karta.....	- 23 -
2.2.2 Úskalí kvalitního zvuku z digitálního zdroje.....	- 23 -
2.2.3 Typy převodníků.....	- 24 -
3 Návrh a realizace generátoru.....	- 31 -
3.1 Návrh.....	- 31 -
3.1.1 WaveStream.....	- 31 -
3.1.2 IWaveProvider.....	- 32 -
3.1.3 Logika.....	- 32 -
3.2 Realizace generátoru.....	- 33 -
4 Ověření vlastností realizovaného generátoru.....	- 37 -
4.1 Porovnání rozdílu signálu určité skupiny zvukových karet.....	- 37 -
4.2 Ověření pomocí integračního a derivačního članku.....	- 40 -
4.3 Závěr ověření vlastností navrženého generátoru.....	- 41 -
5 Manuál k programu generování signálu.....	- 42 -
5.1 Vložení signálu.....	- 42 -
5.2 Zapnutí a vypnutí generování signálu.....	- 44 -

5.3	Práce s vybranými signály.....	- 45 -
5.4	Odebrání všech signálů	- 46 -
5.5	Hlavní nabídka	- 46 -
5.6	Nabídka File	- 46 -
5.7	Nabídka Info.....	- 47 -
Závěr		- 50 -
Použitá literatura		- 51 -
Seznam příloh.....		lii

Úvod

Zvuková karta poskytuje několik způsobů využití, nepočítaje klasickým přehráváním audio záznamu. Další využití je i jako generátor signálu pomocí výstupu, tak pomocí vstupu jako osciloskopu. Vzhledem ke skutečnosti, že zvukovou kartu má v počítači snad každý, rozhodl jsem se pro projekt, ve kterém budu využívat tuto kartu pro generování signálu. Součástí projektu je navržení a realizace programu jako generátoru signálu, s možností si tento program volně stáhnout a používat. Jelikož jsou zvukové karty převážně stereofonní, je program schopen zpracovávat data pro oba kanály současně. Z technického řešení zvukové karty plyne základní omezení, tím je frekvenční rozsah karty.

Obvykle lze zpracovávat a vysílat, či přijímat signály v rozsahu cca 20Hz až 20kHz. U profesionálního (drahého) zařízení by takový rozsah nevyhovoval. Ale díky zvukové kartě máme možnost si jednoduše vyzkoušet a funkce nízkofrekvenčního generátoru, přesto že s omezení. Musím podotknout, že ne každá karta je schopna generovat kvalitní signál pod 20 Hz a totéž platí nad 20kHz, to už jen ty lepší s kvalitnějším filtrem za D/A převodníkem.

Také porovnávám dvě různé možnosti realizace generátoru nízkých frekvencí pomocí dvou knihoven NAudio a CSharp. Obě knihovny jsou pro jazyk C# který má velkou podporu v programátorské komunitě. Zároveň mají silnou podporu i tyto knihovny, které přímo přistupují ke zvukové kartě a předávají parametry ohledně zpracování signálu.

V další části práce se zabývám návrhem a realizací generátoru s časově závislým průběhem, možností nastavení typu průběhu, amplitudy a kmitočtu. Program není třeba instalovat, využívá však .NET a zároveň se snadno instaluje pomocí balíčkovacího NuGet.

Vlastnosti realizovaného generátoru ověřuji pomocí měření a srovnávám s vlastnostmi komerčního generátoru. Taktéž porovnávám výsledky na několika různých třídách zvukových karet. Jak karty integrované, tak ve sběrnici PCI ale i externí (drahé) připojené přes USB s vzorkovací frekvencí 96kHz.

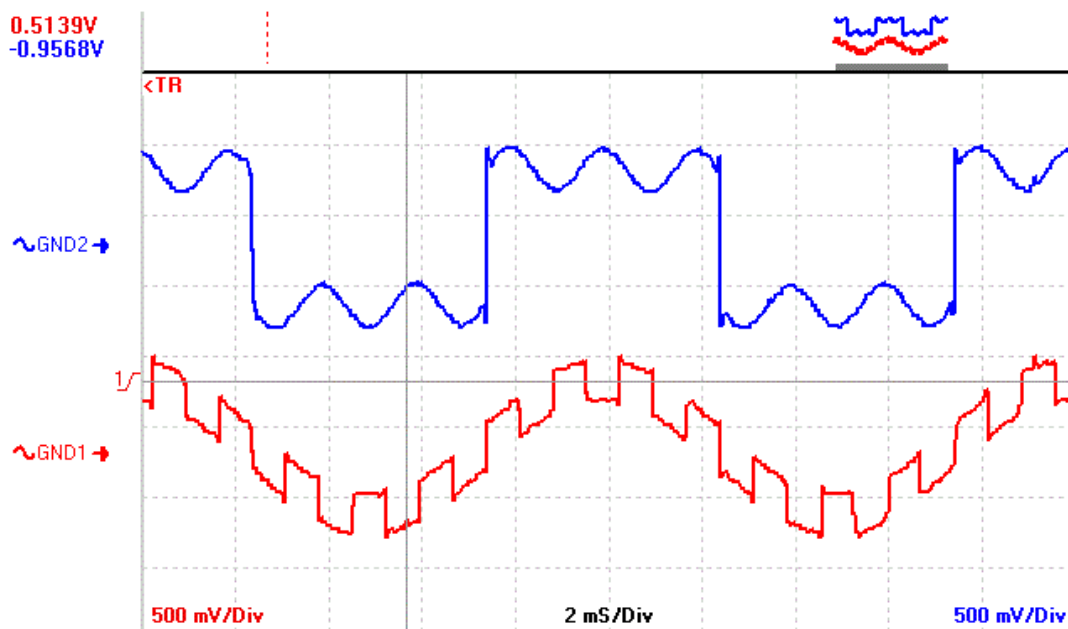
Cílem práce je navrhnout, realizovat a prověřit vlastnosti programovatelného nízkofrekvenčního generátoru časově závislých průběhů s možností nastavit průběh, amplitudu a kmitočtu, nezávisle pro každý kanál zvlášť nebo sdruženě.

1 Možné využití zvukové karty

Je několik způsobů využití zvukové karty počítače pro laboratorní účely, ať už jako osciloskopu využívající vstupu zvukové karty. Nebo jako nízkofrekvenčního generátoru s možností změny amplitudy, frekvence a posunu fáze pro každý kanál zvlášť. Další využití je například v medicíně. Sluch je jedním z našich hlavních smyslů. Diagnostika hloubky jeho poškození využívá zejména audiometrii. Množství standardních zvukových karet má takřka stejné specifikace. Například, vzorkovací frekvence, rozlišení šumu výstupního převodníku (DAC): 44.1 / 48kHz @ 24 bit pro přehrávání. Jsou nabízeny zvukové karty s mnohem vyšší vzorkovací frekvencí například 96kHz. [1]

Architektura počítače a zvukové karty je univerzální, ale to nás může v některých případech limitovat. Například ve chvíli, kdy chceme mít spuštěno velké množství virtuálních efektů či VST instrumentů v reálném čase. V takovém případě se může stát, že počítač zkrátka nestačí plnit naše požadavky. Omezení bývá výpočetní výkon procesoru, propustnost mezi procesorem a pamětmi atd. [8]

V dnešní době je možné si stáhnout z internetu program na generování signálů. O všem naprogramovaný na míru požadavků laboratorního měření je jen jeden. Navržený generátor by měl být schopen měnit amplitudy, na obou kanálech zvlášť či sdruženě, měnit kmitočet neboli frekvenci, a také typ signálu, s možností superponovat například obdélníkový signál na sinusový signál. Nebo naopak superponovat sinusový signál na hradbový signál. Stejně jako je tomu na obrázku 1.1.



Obrázek 1.1: Superponování sinusového signálu na hradbový a naopak

Volba podporované knihovny pro obsluhu zvukové karty

Nabízí se několik variant jak přistupovat pomocí programu ke zvukové kartě pomocí knihoven. Nejrozšířenější možnosti jsou dvě, využití NAudio.dll nebo CSCore.dll. Nyní se zaměřím na krátké srovnání mezi těmito knihovnami.

1.1.1 NAudio

NAudio je open source API pro .NET audio knihovna jenž napsal Mark Heath v jazyku C#, za pomoci příspěvků mnoha autorů. Zároveň je také velmi jednoduché si knihovnu do počítače nainstalovat pomocí balíčkovacího NuGet. Předběžné verze NAudio jsou tak k dispozici. U této knihovny je také možnost stažení několika ukázkových aplikací pro přehrávání, práci s kodeky, práci se zvukovými soubory, manipulace s audio signály a generování zvukových signálů.

Chce-li vývojář úspěšně vyvíjet aplikace, které zpracovávají digitální zvuk, existují některé klíčové pojmy, které musí pochopit. Aby kterýkoliv vývojář rychle pochopil, co potřebuje znát před tím, než se pokusí používat NAudio, je dobré absolvovat kurz nebo si přečíst knihu Digital Audio Fundamentals a nasát tak veškeré informace pokrývající vzorkovací frekvence, bitové hloubky, formáty souborů, kodeky, decibely, ořezávání, aliasing, syntézu, vizualizace, efekty a mnoho dalšího. Konkrétně čtvrtý modul o signálních řetězcích je důležitým balíčkem základních informací.

Nástroj NAudio byl vytvořen, protože knihovna Framework Class dodávaná s Framework.NET 1.0 neměla podporu pro přehrávání zvuku. Název oboru System.Media představený v Framework.NET 2.0 poskytl malé množství podpory a MediaElement ve WPF. Vize společnosti NAudio je poskytovat komplexní soubor tříd souvisejících s audiem, které umožňují snadný rozvoj nástrojů, které hrají nebo zaznamenávají zvuk nebo jakýmkoli způsobem manipulují s audio soubory. NAudio je licencována pod veřejnou licenci společnosti Microsoft (MS-PL), což znamená, že je možno ji používat v libovolném projektu, včetně komerčních projektů.

Rád bych připomenul některé technické funkce NAudio.

- Přehrávání zvuku pomocí různých rozhraní API, WaveOut, WASAPI, DirectSound
- Práce se zvuky mnoha formátů, WAV, AIFF, MP3, G.711, WMA, AAC, MP4
- Převod mezi formami nekomprimovaného zvuku, mono na stereo a naopak, změna bitové hloubky 8,16,24,32 bit.
- Kóduje zvuk pomocí libovolného kodeku ACM nebo Media Foundation, MP3, AAC/MP4WMA, WAV
- Plný model událostí, čtení, odeslání souboru MIDI
- Všechny třídy jsou snadno děditelné pro vlastní program s vlastními komponentami

1.1.2 CScore

CScore je bezplatná knihovna .NET, která je kompletně napsána v C #. Přestože je to stále spíše mladý projekt, nabízí spoustu funkcí, jako je hrát nebo zachytit zvuk, dekodovat nebo mnoho různých kodeků, efektů a mnohem více!

CScore je založen na velmi rozšiřitelné architektuře, která umožní bez značné náročnosti vyhovět většině požadavků. Další možností je vytvořit hudební přehrávače, hlasové rozhovory, zvukové nahrávky a tak dále. Taktéž je velmi jednoduché si knihovnu do počítače nainstalovat pomocí balíčkovací NuGet. Pro zajímavost uvedu ještě pár plusů, jako jsou:

- Vysoce optimalizované performance pomocí CLI
- Navrženo pro nováčky a profesionály
- Rychlá podpora na codeplex nebo stackoverflow
- Vysoké pokrytí kódem pomocí testů jednotky
- Licencováno podle MS-PL (které umožňuje používání cscore pro komerční produkty)
- Podporované funkce jsou velmi podobné jako u NAudio, například:
- Kodeky, MP3, WAVE, FLAC, WMA
- Přehrávání zvuku pomocí různých rozhraní API, WaveOut, WASAPI, DirectSound
- Navíc má podporu 3D zvuku
- Optimalizace pro hry
- Hlasová implementace streamového zdroje

Tato pokročilá zvuková knihovna napsaná v C # poskytuje množství funkcí. Od přehrávání a nahrávání zvuku k dekodování, kódování zvukových toků, souborů ke zpracování audio dat v reálném čase (např. Použití uživatelských efektů během přehrávání, vytváření vizualizací). Možnosti jsou téměř neomezené. [3]

1.2 Volba použité knihovny

Volba mezi NAudio a CScore byla velmi těžká, obě knihovny poskytují kvalitní zázemí a zároveň jsou použity v mnoha projektech. Nakonec jsem se rozhodl na základě velikosti komunity podporující tuto knihovnu. Pro porovnání:

CScore

- Projekt se dělí na 6 větví má 4 vydání a 14 přispěvatelů.

NAudio

- Projekt se dělí na 2 větve, má 8 vydání a 41 přispěvatelů.

Z tohoto důvodu jsem se rozhodl použít knihovnu NAudio, která má silnější komunitu přispěvatelů a zároveň menší počet větví samotného projektu.

2 Signály

Pro laboratorní účely potřebujeme produkovat mnoho různých typů signálu, jako jsou sinusové vlny, čtvercové vlny či obdélníkové vlny, trojúhelníkové vlny, vlnovody a různé pulzy a impulzy a v neposlední řadě šum.

Technicky řečeno, generovaný signál či vlny jsou v podstatě vizuální reprezentací změny napětí v průběhu času. Což znamená, že pokud jsme zaznamenaly tyto změny napětí na kus grafového papíru či osciloskopu, na základně osy x a času T , výsledný graf by představoval tvar vlnové křivky. Ať už je křivka periodická, neperiodická, symetrická, nesymetrická, jednoduchá nebo složitá, všechny křivky zahrnují následující tři společné charakteristiky:

Perioda: je délku času v sekundách, kdy se křivka opakuje od začátku do konce. Tuto hodnotu lze také nazývat periodický čas T . Například sinusový signál nebo šířka impulsu pro hradbový signál.

Frekvence: je počet opakování průběhu vlny během jedné sekundy. Frekvence je reciproční časové období, se standardní jednotkou frekvence Hertz, (Hz).

Amplituda: je maximální hodnota periodicky měnící se veličiny, v našem případě se jedná o řády milivoltů. [7]

2.1 Periodický signál

Periodické signály jsou nejběžnější ze všech průběhů, jelikož zahrnují sinusové vlny. Střídavý průběh střídavého proudu v každé domácnosti je sinusová vlna, která neustále kmitá mezi maximální a minimální hodnotou v průběhu času.

Množství času mezi jednotlivými opakováními cykly sinusového tvaru vlny je známo jako "periodický čas" nebo jednoduše "perioda". Jinými slovy, čas potřebný k tomu, aby se křivka opakovala.

Tato doba se může měnit u každé vlny od zlomku mikrosekund, protože závisí na frekvenci průběhu. Například sinusový průběh, který trvá jednu vteřinu pro dokončení cyklu, bude mít periodu 1 sekundy. Podobně i sinusová vlna, která trvá pět sekund, bude mít periodu pěti sekund a tak dále.

Takže jestliže délka času, který vyžaduje, aby průběh dokončil jeden úplný cyklus, než se sám opakuje, je nazýván "periodou vlny" a měří se v sekundách, můžeme pak vyjádřit průběh jako číslo období na druhý označený písmenem T , jak je znázorněno na obrázku 1.2.

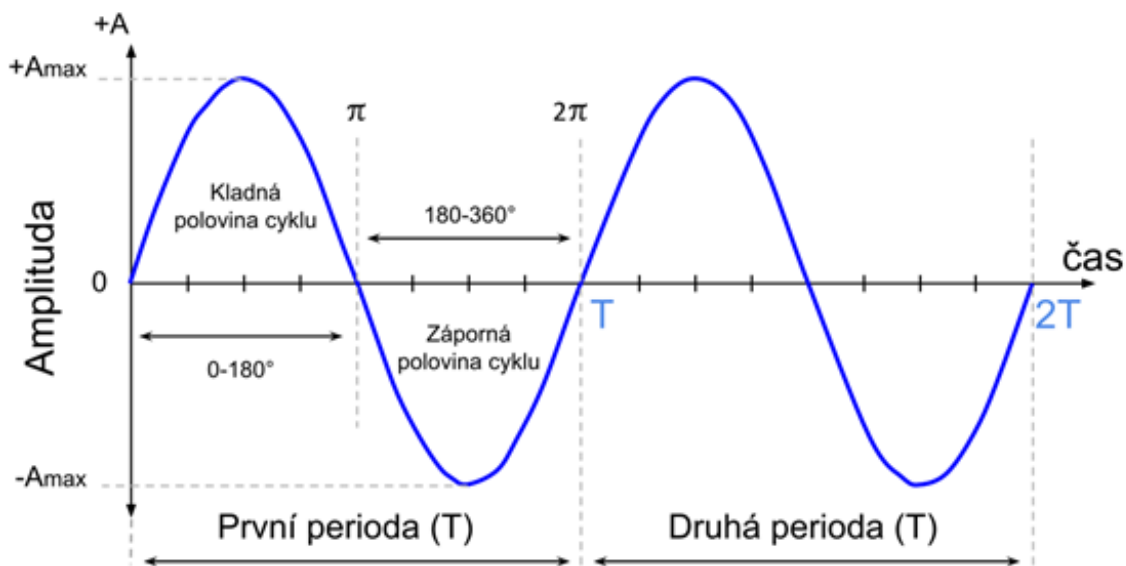
Střídavý průběh je definován jako periodický průběh, jeho plochy omezené křivkou nad časovou osou T a pod ní jsou stejné. Tvar může mít harmonický sinusový nebo neharmonický, ten se dá rozložit podle Fourierovy transformace na součet harmonických složek. Tyto složky mají frekvenci, která je celistvým násobkem základní harmonické frekvence. [2]

2.1.1 Sinusový signál

Sinusoidy hrají důležitou roli ve zpracování signálu z následujících důvodů:

- lze je jednoduše generovat
- snadno se s nimi pracuje
- jejich matematické vlastnosti jsou dobře známy
- všechny ostatní signály mohou být vytvořeny jako součet sinusoid (Fourierovy řady)
- Ještě chci připomenout, že spojitě sinusové signály s různou frekvencí jsou vždy různé. A také s rostoucí frekvencí roste počet oscilací sinusovky v daném časovém intervalu.

Pouze pro sinusové vlny můžeme vyjádřit periodický čas vlnového tvaru v libovolných stupních nebo radiánech, protože jeden plný cyklus se rovná 360° ($T = 360^\circ$) nebo Radians jako 2π , 2π ($T = 2\pi$), pak můžeme říci, že 2π radians = 360° . Jak je známo, doba potřebná k tomu, aby se křivky opakovaly, je známá jako periodický čas nebo perioda, která představuje pevnou dobu. Pokud vezmeme reciproční období ($1 / T$), dostaneme hodnotou, která označuje, kolikrát se opakuje perioda nebo cyklus v jedné sekundě nebo v cyklech za sekundu a to je obecně známé jako Frekvence s jednotkami Hertz, (Hz) . Pak lze také definovat Hertz jako "cykly za sekundu" (cps) a 1Hz přesně odpovídá 1 cyklu za sekundu.



Obrázek 2.1: Sinusová vlna

Vztah mezi frekvencí a periodickým časem, kde: f je v Hetzích a T v sekundách.

$$f = \frac{1}{T} \text{ Hz} \quad (2.1)$$

$$T = \frac{1}{f} \text{ sec} \quad (2.2)$$

Sinusoidy hrají důležitou roli ve zpracování signálu z následujících důvodů:

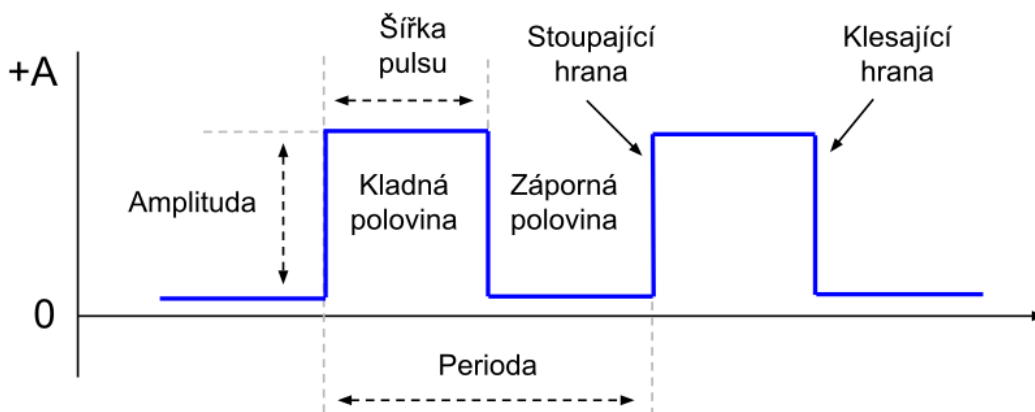
- lze je jednoduše generovat
- snadno se s nimi pracuje
- jejich matematické vlastnosti jsou dobře známy

Všechny ostatní signály mohou být vytvořeny jako součet sinusoid (Fourierovy řady) odezva lineárních systémů na harmonický vstup je vždy harmonická se stejnou frekvencí jako je vstup. (tzv. sinus fidelity) Ještě chci připomenout, že spojitě sinusové signály s různou frekvencí jsou vždy různé. A také s rostoucí frekvencí roste počet oscilací sinusovky v daném časovém intervalu. Střídavý průběh je definován jako periodický průběh, jeho plochy omezené křivkou nad časovou osou t a pod ní jsou stejné. Tvar může mít harmonický – sinusový, nebo neharmonický – ten se dá rozložit podle Fourierovy transformace na součet harmonických složek. Tyto složky mají frekvenci, která je celistvým násobkem základní harmonické frekvence. [9]

2.1.2 Hradbový signál

Křivky vlnových délek se používají v elektronických obvodech pro řízení hodin a časování, neboť jsou to symetrické průběhy stejného hradbového či čtvercového trvání. Téměř všechny digitální logické obvody používají na svých vstupech a výstupech křivky s obdélníkovými vlnami.

Na rozdíl od sinusových vln, které mají hladký průběh vzestupu a pádu se zaoblenými rohy při jejich pozitivních a záporných špičkách, mají čtvercové vlny na druhou stranu velmi strmé, téměř svislé směrem nahoru a dolů s plochou horní a spodní částí, - "čtverce", jak je znázorněno na obrázku 2.2



Obrázek 2.2: Vlna čtvercového signálu

Víme, že signály ve tvarech čtverců mají symetrický tvar, jelikož každá polovina cyklu je identická, takže doba, která je kladná, musí být stejná jako doba, kdy je šířka pulsu záporná nebo nula.

Pak můžeme říci, že pro vlnový průběh a čas z kladné části obdélníkového nebo čtvercového signálu je roven záporné polovině. Vzhledem k tomu, že frekvence se rovná vzájemnému období ($1 / T$), můžeme definovat frekvenci křivky s obdélníkovou vlnou jako:

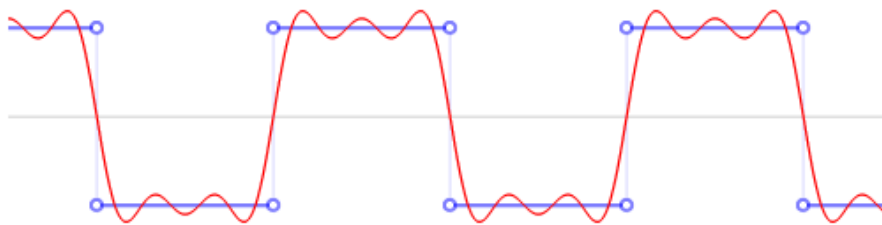
$$f = \frac{1}{KT+ZT} \text{ Hz} \quad (2.3)$$

Kde: f je v Hetzích a KT je čas Kladné poloviny signálu a ZT je čas záporné poloviny. Obdélníkový průběh křivek je symetrický tvar a má pozitivní šířku impulsu rovna jeho negativní šířce impulsu. Křivky s obdélníkovými vlnami se používají v digitálních systémech k reprezentování logické úrovně "1" vysoká amplituda a logické úrovně "0", nízká amplituda.

Fourierova řada může být použita k vyjádření funkce z hlediska frekvencí (harmonických), ze kterých je složena. Když budeme sčítat donekonečna sinusové vlny podle vzorce:

$$U(t) = \frac{\sin(2\pi.f_0.t)}{1} + \frac{\sin(2\pi.f_0.t)}{3} + \frac{\sin(2\pi.f_0.t)}{5} + \frac{\sin(2\pi.f_0.t)}{7} + \dots \quad (2.4)$$

bude signál stále hranatější a nakonec získá obdélníkový tvar, viz obrázek 2.3 a 2.4.

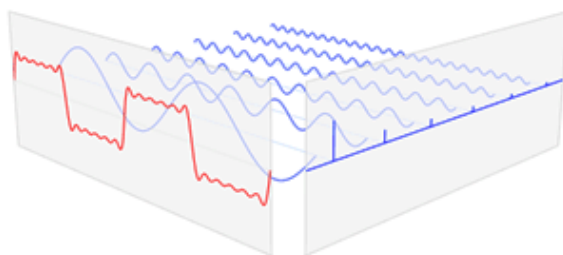


Obrázek 2.3: $N=3$



Obrázek 2.4: $N=6$

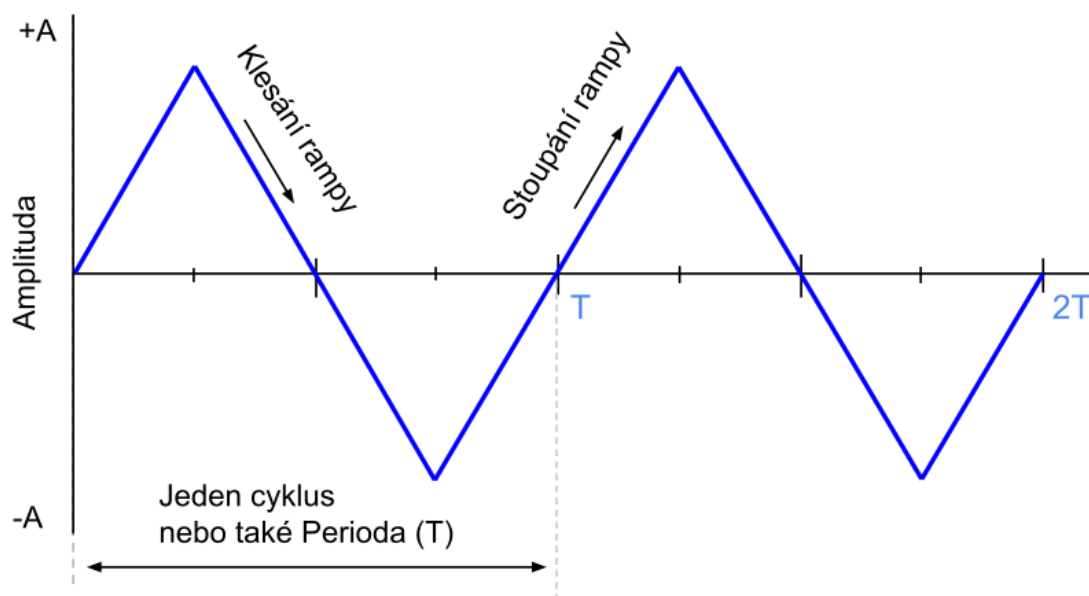
Proto obsahuje spektrum obdélníkového signálu nekonečné množství vyšších harmonických, viz obrázek 2.5



Obrázek 2.5: Znáznornění vyšší harmonické ve Fourierově řadě

2.1.3 Trojúhelníkový signál

Trojúhelníkové vlny signálu jsou obecně obousměrné nesinusové vlnové kmity, které kmitají mezi kladnou a zápornou špičkovou hodnotou. Trojúhelníková vlna je ve skutečnosti více symetrická lineární rampa, neboť jde pouze o pomalý stoupavý a klesající průběh signálu. Rychlost, se kterou se mění napětí mezi každým směrem rampy, je stejná v obou polích cyklu, jak je znázorněno na obrázku. [9]



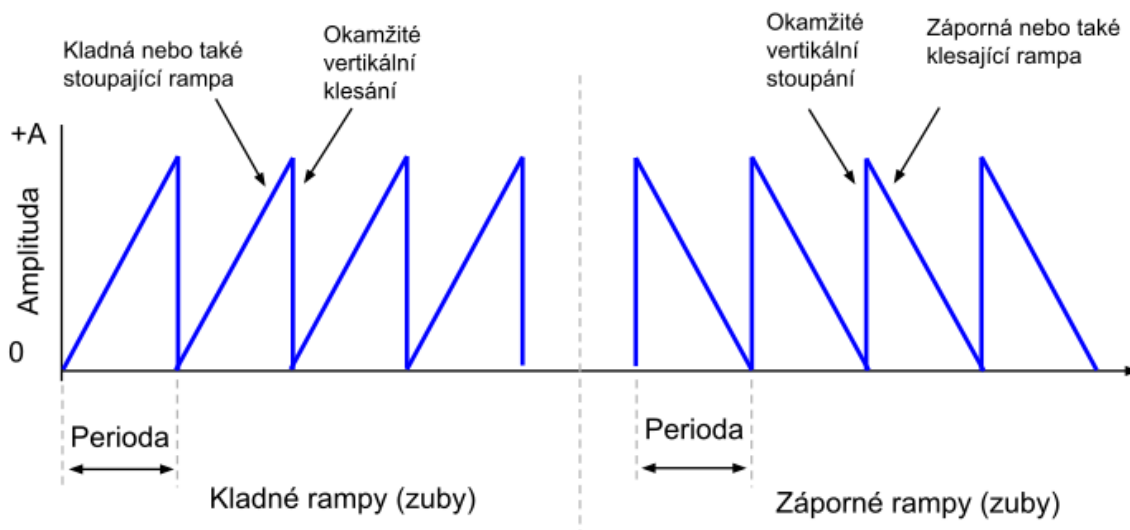
Obrázek 2.6: Vlna trojúhelníkového signálu

Obecně platí, že pro trojúhelníkové průběhy je doba klesání rampy stejná jako stoupání rampy, která dává trojúhelníkovým průběhům 50% pracovní cyklus.

Takže pro pomalý vzestup a pomalý zpoždovací čas rampy bude nižší průměrná úroveň napětí než rychlejší doba vzestupu a rozpadu. Mohli bychom však produkovat nesymetrické trojúhelníkové průběhy změnou buď stoupajících, nebo rozkládajících se hodnot ramp, aby nám poskytl jiný typ vlnového tvaru, který je běžně znám jako pilové zuby. [2]

2.1.4 Pilové signály

Kladná rampová vlnová tvarovka je běžnějším ze dvou typů vln, přičemž rampová část vlny je téměř dokonale lineární. Křivka pilových zubů je obvykle dostupná u většiny generátorů funkcí a skládá se ze základní frekvence (f) a všech jejích celočíselných poměrů rovných harmonických pouze, $1/2$, $1/4$, $1/6$, $1/8$... $1/n$ atd. V praxi to znamená, že vlny pilový zubů jsou bohaté na harmonické hudební syntezátory a hudebníkům dávají kvalitní zvukové nebo tónové barvy, bez jakéhokoli zkreslení. Názorná ukázka pilových signálů je na obrázku 2.7. [9]



Obrázek 2.7: *Pilové signály*

2.2 Vzorkování signálů

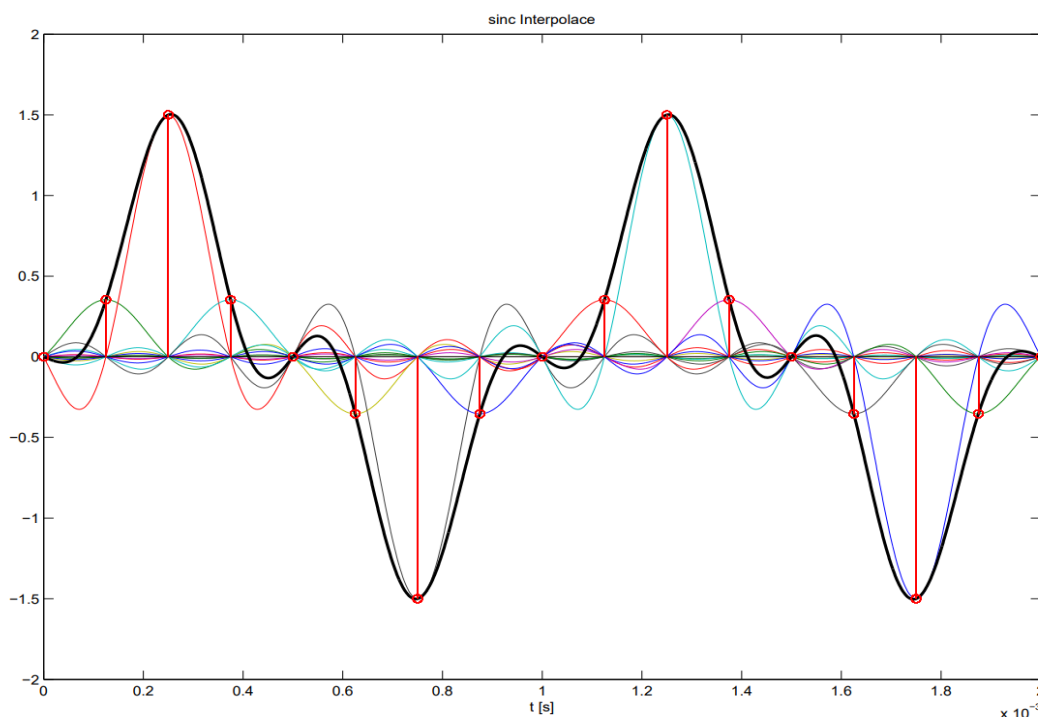
Vzorkování je proces, při němž dochází k převodu analogového signálu na posloupnost diskrétních vzorků, které s dostatečnou přesností charakterizují původní analogový signál. Pro zdařilou rekonstrukci analogového signálu z posloupnosti diskrétních vzorků musí být mezi vzorkovací frekvencí a maximální frekvencí vzorkovaného signálu splněn vztah (vzorkovací teorém, Shannonův-Kotělnikovův teorém)

$$f_{vz} = 2f_N \geq 2f_{Max} \quad (2.5)$$

kde f_{Max} je maximální frekvence obsažená ve vzorkovaném signálu a f_N je Nyquistova frekvence, tedy nejvyšší frekvence, kterou je možné při dané vzorkovací frekvenci správně navzorkovat. Spektrum navzorkovaného signálu je periodické s periodou f_{vz} . Odpovídá spektru analogového signálu na frekvencích $k \cdot f_{vz}/N$, $k \in \mathbb{Z}$. Interval $\Delta f = f_{vz}/N$ nazýváme frekvenčním rozlišením. Proces vzorkování lze popsat vztahem

$$x[n] = \sum_{n=-\infty}^{\infty} x(nT_{vz})\delta(t - nT_{vz}) \quad (2.6)$$

Názorně tento princip vysvětluje obrázek 7. V každém vzorkovacím okamžiku nT_{vz} se vytvoří průběh funkce $x[n]\text{sinc}(t - nT_{vz})$ s maximální amplitudou rovnou hodnotě vzorku. Výsledný signál je pak dán součtem jednotlivých průběhů $x[n] \text{sinc}(t - nT_{vz})$. [3]



Obrázek 2.8: Interpolace funkcí $\text{sinc}()$

2.2.1 USB DAC, aneb externí zvuková karta

DAC (Digital to Analog Converter), tedy česky *D/A převodník* je elektronická součástka pro převod digitálního signálu na analogový a obráceně. Nazývá se tak ale i zvuková karta k počítači s primárním zaměřením na kvalitní převod digitálního signálu z počítače na analogový, u níž je DAC jedna z nejpodstatnějších částí. Na rozdíl od normálních, herních, studiových či jiných zvukových karet nemívá funkcionalitu typu vícekanálový zvuk, vstupy nebo laškovné audio efekty. Je to náhrada CD přehrávače, magnetofonu či gramofonu.

Co se funkcí týče, jsou převodníky vybaveny střídmě. Některé mají zeslabovače, u jiných se intenzita signálu ovlivňuje pouze digitálně. Malé DACy mají obvody uvnitř obvykle na jednom tištěném spoji. Vnitřní uspořádání a zvolené součástky ovlivňují charakter zvuku. Lze postavit i USB DACy z velice drahých komponent, jejichž kvalita zvuku stoupá společně s cenou, která z desítek tisíc přesahuje občas i do stovky tisíc.

Za srdce DACu lze považovat samotný D/A převodník, který ovlivňuje kvalitu zvuku ale také podporované datové toky (PCM, PDM) nebo jejich bitovou hloubku a frekvenci.

bitová hloubka – určuje rozlišení signálu, počet bitů pro vyjádření informace o intenzitě zvuku. Čím vyšší, tím větší dynamika. Kvalitní nahrávky dnes mají 24bit a ačkoliv běžná CD mají 16bit, lze zbylou hloubku použít pro snížení intenzity zvuku bez zkreslení.

podporované vzorkovací frekvence – nejběžnější je sice 44,1kHz, na které v počítači běží většina programů, videí a také hudby, avšak kvalitní, ale i třeba studiové nahrávky bývají ve vyšších formátech (48, 96). Formát DVD-Audio či některé online obchody nabízí hudbu i v 192kHz. Pokud převodník podporuje danou vzorkovací frekvenci nahrávky, dokáže ji přehrát bez podvzorkování.

2.2.2 Úskalí kvalitního zvuku z digitálního zdroje

Aby DAC vytvářel zvuk, s co největší přesností záleží na více faktorech. Jeden z nejpodstatnějších je propojení zařízení s počítačem. K tomu se nejčastěji používá USB, které přináší hned několik komplikací: Jednak má různé verze s rozdílnými parametry ale také ve spolupráci s ostatními prvky v počítači znehodnocuje signál přidáním tzv. jitteru (chvění).

Verze USB nejsou až takový problém, protože jich není mnoho a i již velmi stará verze 1.1 (tzv. Full-Speed USB) stačí pro přehrávání signálu ve 24 bitové hloubce při frekvenci 96kHz, přičemž v dnešní době se prodávají počítače i notebooky s USB 3 a starší USB 2 (High-Speed USB) podporují i víceleté stroje. Poněkud komplikovanější to je s USB audio ovladači. USB DACy totiž spoléhají na standardizované ovladače USB Audio Class, které existují ve

dvou verzích. Starší verze 1.0 vznikla pro původní USB a její maximální přenosovou rychlostí je 12 Mbit/s (USB 1.1), do které se vejde výše zmíněný signál 24bit/96kHz. Tato verze je podporována systémy Windows XP a vyšší, Linux a OSX Snow Leopard a vyšší. Novější verze 2.0 je zpětně kompatibilní a využívá vyšší propustnost (dle USB 2.0 specifikace teoreticky 480 Mbit/s, prakticky tak půlku), která je potřebná pro vyšší datové toky, např. 24bit/192kHz. Ačkoliv tuto verzi podporuje Linux i OSX, Microsoft ovladač neimplementoval a výrobci DACů tak musejí implementovat ovladače vlastní.

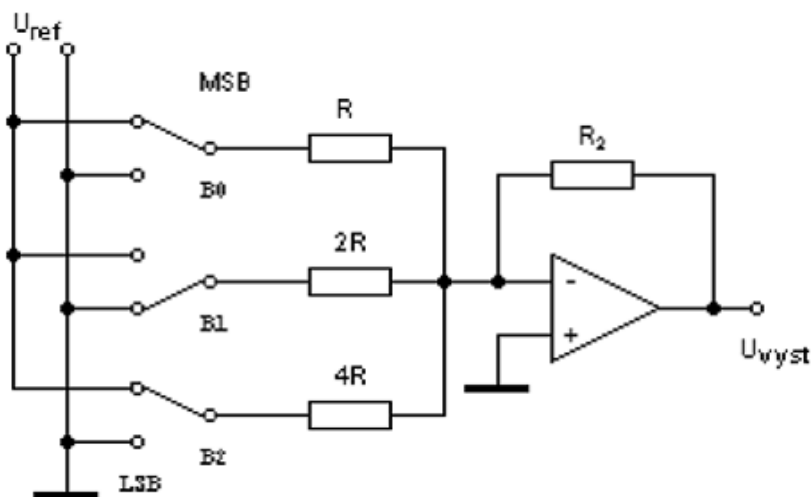
2.2.3 Typy převodníků

- Přímé - vstupní datové slovo je přímo převedeno na výstupní napětí, příp. proud.
- Nepřímé - Pomocný signál má tvar impulzu, měronosnou veličinou je šířka impulzu konstantní amplitudy, příp. poměr šířky impulzu k době převodu (střída)

Přímé – s váhovou strukturou odporové sítě – napět'ové váhování

- s příčkovou strukturou odporové sítě $R - 2R$
- s váhovou strukturou kapacitorové sítě – nábojové váhování kombinované
- Sériové převodníky
 - S nábojovou redistribucí
 - Algoritmický

Princip D/A převodníku váhovou strukturou odporové sítě je znázorněn na obr 2.9.



Obrázek 2.9: D/A převodník s váhovou strukturou odporové sítě

Řídící veličinou je vstupní datové slovo, předávané prostřednictvím datové sběrnice. Jednotlivé bity vstupního slova s váhou $2^0, 2^1, 2^2, \dots, 2^n$ ovládají jednotlivé elektrické přepínače, které mají v sérii odpory o takových hodnotách, že každá další hodnota je vždy dvojnásobkem předcházející. Pro napětí na výstupu převodníku můžeme psát:

$$U_{vyst} = -U_{ref} \frac{R_2}{R} \sum_{i=0}^n \frac{B_i}{2^i} \quad (2.7)$$

kde n je počet bitů datového slova a B_i nabývá hodnot 0 nebo 1 podle stavu příslušného spínače. [11]

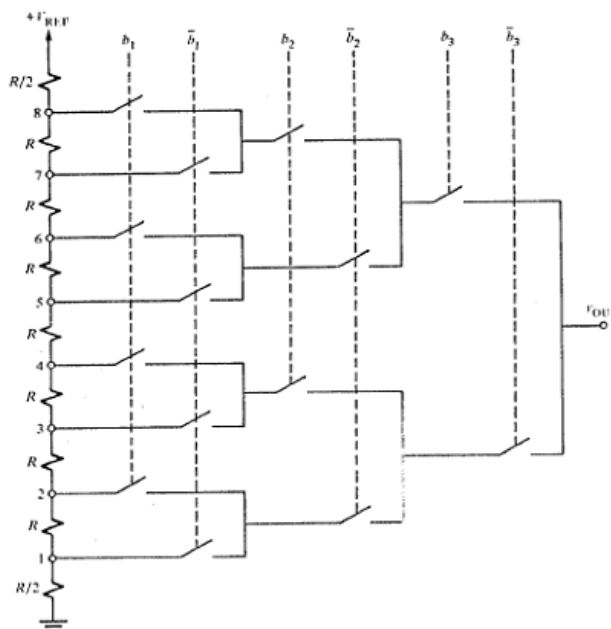
Výhody:

- Rychlost
- Monotoničnost

Nevýhody:

- Přesnost - velký rozsah hodnot odporů
- Malý počet bitů
- Nemožnost vyrobit tak přesné hodnoty rezistorů

Velký rozsah hodnot odporů lze vyřešit zapojením na obr. 2.10. Kde se vyskytují jen dvě hodnoty odporu. Jejichž poměr je 2:1 a rezistory v poměru se dají vyrobit s daleko větší přesností.



Obrázek 2.10: D/A převodník s napětovým váhováním

V tomto zapojení se využívá resistorů zapojených mezi referenční napětí a zem k dosažení váhovaných hodnot napětí mezi těmito hodnotami. Pro N -bitový převodník bude potřeba 2^N segmentů. Tyto segmenty jsou připojeny do přepínacího stromu, kde jsou spínače ovládány bitovým slovem. Jestliže je i bit 1, pak spínače řízeny bitem b_i jsou sepnuty. Jestliže je i bit 0 pak jsou sepnuty spínače b_{ineg} .

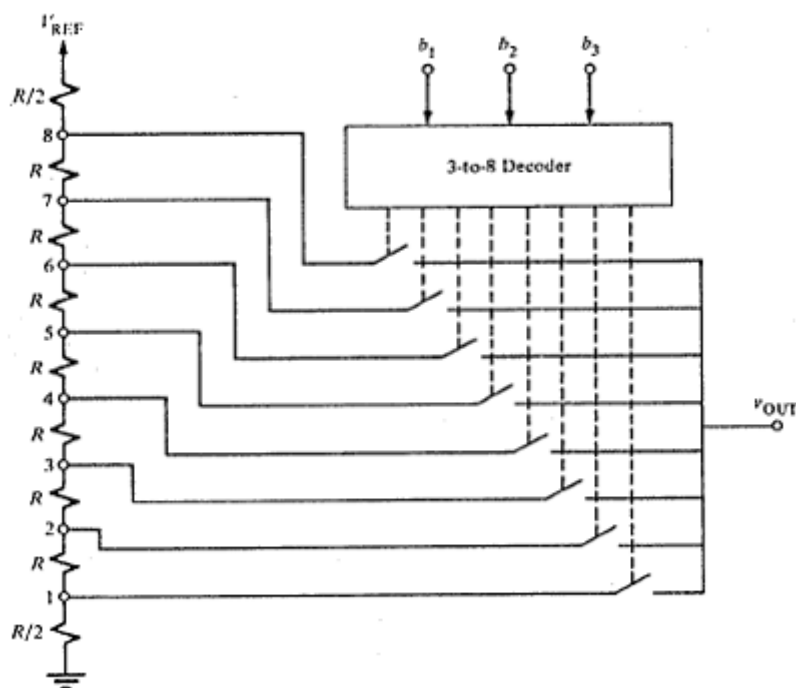
Výhody:

- Rychlost
- Monotoničnost

Nevýhody:

- Velký počet odporů a spínačů pro větší počet bitů
- Díky tomu omezení na cca. 8bitů
- Parazitní vlastnosti spínačů
-

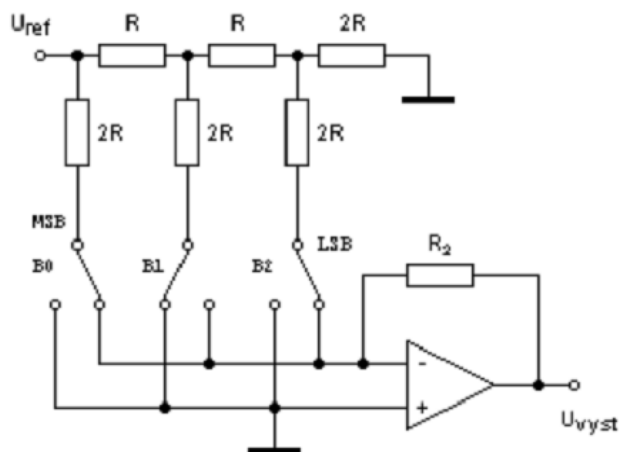
Parazitní vlastnosti spínačů se dají částečně redukovat snížením počtu spínačů a použitím dekodéru N to 2^N .



Obrázek 2.10: D/A převodník s napěťovým váhováním s menším počtem spínačů

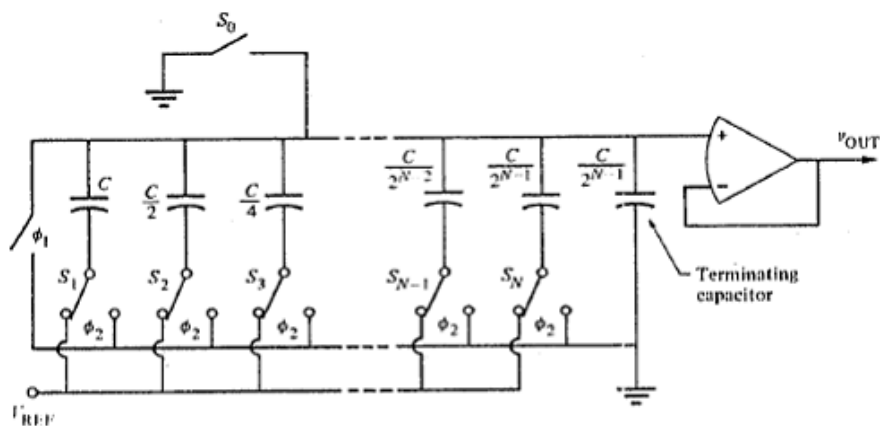
Ostatní vlastnosti jsou stejná jako v předchozím zapojení.

Jako další bych rád představil D/A převodník s příčkovou strukturou odporové sítě na obrázku 2.11. Vstupní proud z referenčního zdroje napětí se dělí v každém uzlu a odpovídá dvojkové váze. Jelikož odpory mají stejnou, resp. srovnatelnou hodnotu, mohou být vyrobeny stejnou technologií, čímž dosáhneme snadněji jejich stejnou toleranci a teplotní závislost. Struktura odporové sítě je uspořádána tak, že příspěvek každého následujícího bitu nalevo od každého uzlu je R . V důsledku toho příspěvek následujícího bitu k výstupnímu analogovému napětí se vždy zmenšuje s násobkem 0,5. Odstraňuje nevýhody předchozích zapojení s množstvím součástek a rozsahem velikostí součástek. [11]



Obrázek 2.12: D/A převodník s příčkovou strukturou odporové sítě

D/A převodníky s váhovou strukturou kapacitorové sítě – nábojové váhování. Funguje na principu binárního dělení náboje v kapacitorové síti. Převodník pracuje ve dvou fázích.. Během fáze jedna jsou spodky kapacitorů připojeny na zem. Dále během fáze dva, jsou spínače s binární hodnotou 1 připojeny na referenční napětí. Výsledná situace může být popsána jako poměr náboje v kapacitorech připojených na referenční napětí ku celkovému náboji. Jako je na obrázku 2.12



Obrázek 2.12: Nábojové váhování

D/A převodník s nábojovým váhováním funguje na principu binárního dělení náboje v kapacitorové síti. Převodník pracuje ve dvou fázích.. Během fáze jedna jsou spodky kapacitorů připojeny na zem. Dále během fáze dva, jsou spínače s binární hodnotou 1 připojeny na referenční napětí. Výsledná situace může být popsána jako poměr náboje v kapacitorech připojených na referenční napětí k celkovému náboji.

Výhody:

- Rychlost

Nevýhody:

- Velký poměr hodnot kapacitorů, omezení na cca 1024:1 tj. 10 bitů
- Nemonotoničnost,

Kombinované D/A převodníky

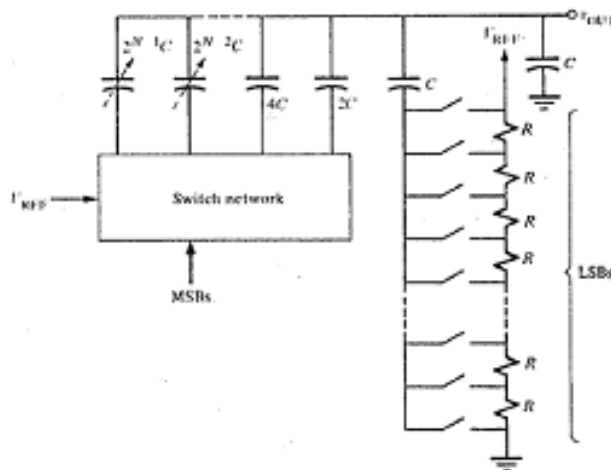
Schéma na obrázku 2.13, odpovídá MSB bitům. Bit 7 odpovídá 1/128 referenčního napětí. Schéma na obrázku 2.14 pak odpovídá LSB bitům bit 8 nabije dělicí kapacitu na $\frac{1}{2}$ a proto má hodnotu $\frac{1}{2}$ bitu 7.

Výhody:

- Nízký rozptyl hodnot kapacitorů a tudíž nižší spotřebovaná plocha.

Nevýhody:

- Dělicí kapacitor musí být velice přesný
- Nemonotoničnost

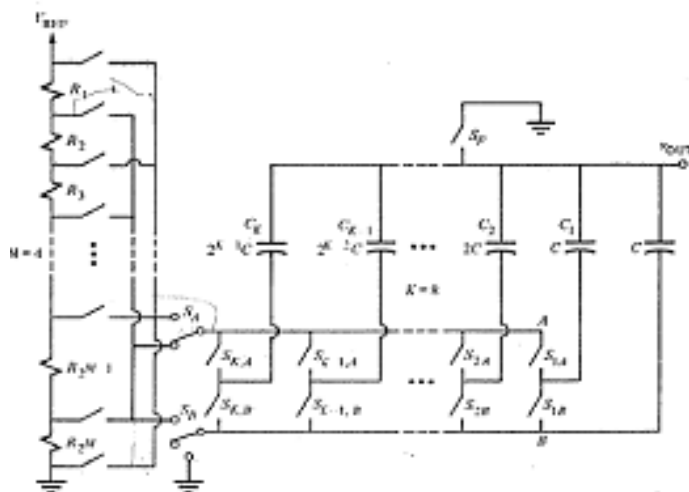


Obrázek 2.13: Převodník MSB

U prvního případu na obrázku 2.13 probíhá nejdříve napěťové dělení pomocí rezistorové sítě a pak následuje nábojové dělení pomocí kapacitorové sítě. Na druhém obrázku 2.14 je tomu opačně. [11]

Výhodou může být relativně vyšší počet bitů a monotoničnost.

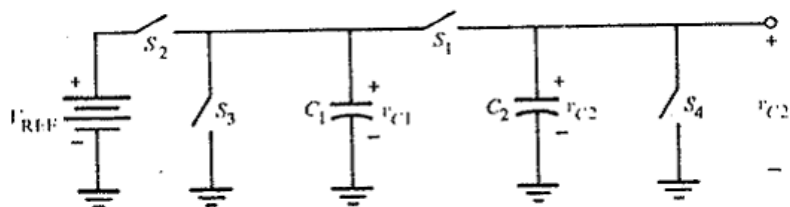
Nevýhodou opět potřeba přesných součástek



Obrázek 2.14: Převodník LSB

Sériové převodníky

Jsou takové převodníky, u nichž konverze probíhá odděleně. Obvykle jeden hodinový pulz na jeden bit. Na obr 2.15 je seriový převodník s nábojovou redistribucí. Pouze 4 spínače a 2 kapacitory jsou potřeba pro tento druh převodníku. Spínač S4 vybíjí kapcitor C2. S2 nabíjí C1 na V_{ref} a S3 vybíjí kapcitor C1. Spínač S1 je nedistribuční spínač a dělí součet nábojů na kapacitoru na dva stejné díly. Konverze probíhá v N krocích a začíná se od LSB. Jestliže má bit hodnotu 0 kapcitor se vybije a jestliže má hodnotu 1 kapcitor se nabije na V_{ref} . V dalším kroku je sepnut pouze spínač S1 a náboj rozdělí na dvě poloviny. Na C2 se tak objeví $\frac{1}{2}$ V ref. Protože ale za dobu převodu bude spínač sepnut ještě (N-1)krát, bude ještě (N-1)krát půlena hodnota na kapacitoru C2 Pro $N=8$ a $LSB=1$ je na konci převodu na C2 díky jedničce u LSB hodnota $1/2^N$. Druhý nejméně významný bit, jelikož se do převodu zapojil až v druhém kroku, stihne se podělit pouze (N-1)krát a jeho přínos tedy je $1/2^N$.



Obrázek 2.15: Sériový převodník s nábojovou redistribucí

Jako poslední bych rád zmínil nepřímé převodníky. U nepřímých D/A převodníků má buď pomocný signál tvar impulzu, měronosnou veličinou je šířka impulzu konstantní amplitudy, příp. poměr šířky impulzu k době převodu (střída) - převodníky s pulzně šířkovou modulací (PWM – Pulse Width Modulation) nebo je pomocný signál tvořen skupinou impulzů, měronosnou veličinou je počet impulzů konstantní šířky a amplitudy během doby převodu – převodníky s modulací hustotou pulzů (PDM – Pulse Density Modulation).

Nejrozšířenější převodníkem tohoto typu je delta-sigma D/A převodník, jehož zjednodušené blokové schéma je na obrázku 2.16.

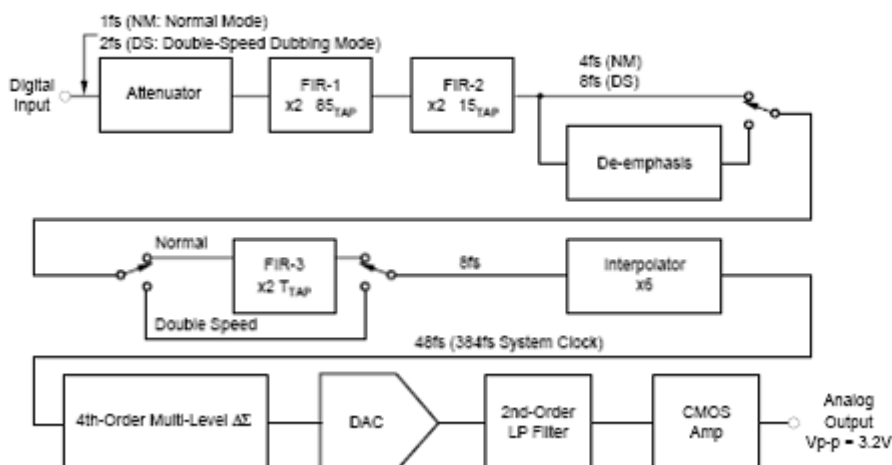
Jak je vidět, převodníky tohoto typu jsou poměrně složité. Zkráceně se dá říct, že se vstupní slovo převede do delta sigma formátu a ten se převede do analogové podoby. [11]

Výhody:

- Rozsah. Velké množství bitů

Nevýhody:

- Složitost



Obrázek 2.16: *Blokové schéma delta sigma D/A převodníku*

3 Návrh a realizace generátoru

Ze základního principu fungování generátorů signálu je jasné, že k realizaci softwarového nízkofrekvenčního generátoru bude zapotřebí pracovat s hodnotami: vzorkovací frekvence, amplituda, kmitočet, obsluha každého kanálu zvlášť nebo společně a typ vlny. Těchto pět klíčových funkcí tvoří jádro programu. Tyto funkce předají své hodnoty objektu `SIGEN_logic` jenž obsahuje `SineWavProvider32` který vypočítá hodnoty matice která je následně odeslána na zvukovou kartu jako signál na výstupu.

3.1 Návrh

Vzhledem k předchozímu zvážení a výběru použití knihovny `NAudio` k zajištění chodu generátoru signálu, bylo nutno nastudovat dokumentaci k této knihovně. Po přečtení dokumentace jsem již věděl které metody použít pro realizaci funkcí signálního generátoru. <https://github.com/naudio/NAudio/tree/master/Docs> Nejdůležitější objekty které je potřeba k realizaci jsou: `IWaveProvider` a vlastnost `WaveFormat`. `WaveFormat` si můžeme představit jako dvourozměrné pole, kdy liché sloupce představují data pro levý kanál a sudé pro pravý.

3.1.1 WaveStream

Základní objekt knihovny `NAudio` je `WaveStream`, tento objekt je děděný z `System.IO.Stream`. `WaveStream` reprezentuje stream audiodat ve formátu, který může být determinován pomocí `WaveFormatu`. [10]

```
{  
buffer[n + offset] = GetWaveTypeFormula(LeftSignalType, sample,  
FrequencyLeft, AmplitudeLeft, sampleRate);  
buffer[n + offset + 1] = GetWaveTypeFormula(RighthSignalType,  
sample, FrequencyRight, AmplitudeRight, sampleRate);  
}
```

Objekt `WaveFormat` má metodu `Read` která má zápis:

```
int Read(byte[] destBuffer, int offset, int numBytes)
```

- Do buffer se vkládají data z `WaveFormatu`.
- Offset zařizuje posun signálu, tento parametr je většinou nastavený na hodnotu 0.
- V `numBytes` je uložen index vzorku signálu

Metoda `Read` je volána ve smyčce a v této metodě se počítá samotný průběh signálu.

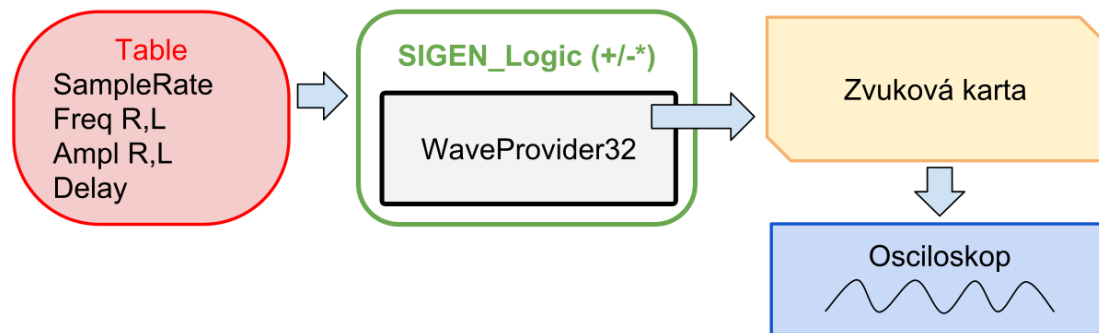
3.1.2 IWaveProvider

Samotná implementace WaveStreamu je velmi náročná. A proto má knihovna objekt IWaveProvider. Tento objekt nám usnadňuje práci, zároveň je s ním práce jednodušší. Její interface je mnohem menší než u metody WaveProvider. V podstatě má jen jednu metodu a tou je Read a jednu vlastnost WaveFormat.

```
public interface IWaveProvider
{
    WaveFormat WaveFormat { get; }
    int Read(byte[] buffer, int offset, int count);
}
```

3.1.3 Logika

Ze schématu je pro představu jasné, jak funguje generování signálu od předání informací jakožto parametrů a hodnot, až po měření a ověření funkčnosti na osciloskopu.



Obrázek 3.1: Schéma fungování zpracování informací pro zvukovou kartu

3.2 Realizace generátoru

Princip, na kterém bude generátor signálu fungovat, je znázorněn na obrázku 3.1. Pro samotnou realizaci signálního generátoru, je využito grafického prostředí Microsoft Visual Studio 2017 a jazyku C#. Ze zadání plyne jasný požadavek na možnost ovládání každého kanálu zvlášť. Jako první se mi vybavil formulář s volbou hodnot jakou je volba kanálu, dále posuvník určující frekvenci a amplitudu, v neposlední řadě typ signálu a tabulka s řádky reprezentující

signál, ve kterých budou sloupce s hodnotami pro každý z těchto signálů. Případná náročnost na vytvoření nastavení generátoru, může být relativně časově zdlouhavá, rozhodl jsem se zahrnout funkcionalitu pro uložení a následné otevření projektu. Projekty se ukládají do souboru ve formátu XML, což přináší snadné nahlédnutí a případnou editaci mimo program. Aby bylo ovládání programu jednoduché a skoro až intuitivní, použil jsem jednoduchý grafický interface v anglickém jazyce. Který je rozdělený do dvou částí, v horní části tabulku se znázorněním nastavení signálů a snadné editace. Pořadí sloupců v tabulce je seřazeno podle důležitosti hodnoty, jako je například frekvence a její častá editace. Ve spodní části jsou ovládací prvky rozděleny

do čtyř sekcí, v první sekci je prvek pro přidání signálu (single/multi), v druhé sekci jsou ovládací prvky pro ovládání stavu generátoru (zapnout/vypnout) a v třetí sekci jsou prvky pro práci

se signály. V poslední čtvrté sekci je prvek pro rychlé odstranění všech přidávaných nastavení signálů. Takže po vytvoření hlavního formuláře bylo potřeba vytvořit ještě formulář pro zadávání nových nastavení generovaných signálů. Vytvoření toho formuláře nebylo nějak náročné, počet prvků pro nastavení nového signálu, je dán počtem sloupců v tabulce na hlavním formuláři.

Po vytvoření grafického prostředí bylo potřeba přejít k samotnému programování generátoru signálu. Jak uvádím v předešlé kapitole, pro usnadnění práce je využito třídy IWaveProvider. Která je rodičem nově vyděděné třídy WaveProvider32, tato nově vzniklá třída přenastaví hodnoty a přepíše metody v již existujícím IWaveProvideru. Ukázka dědění třídy IWaveProvider je možná vidět na obrázku 3.2. [4]

```

public abstract class WaveProvider32 : IWaveProvider
{
    private WaveFormat waveFormat;

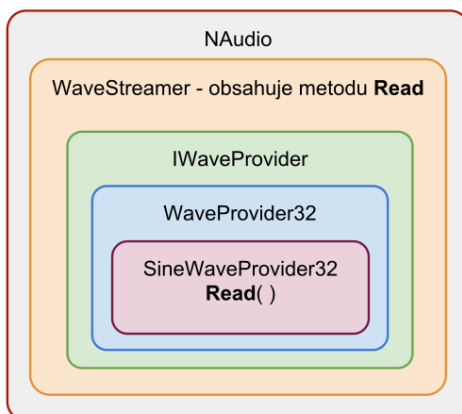
    Počet odkazů: 0 | zbynek fojtik, Před 84 dny | 1 autor, 1 změna
    public WaveProvider32()
        : this(44100, 2) /*Means two channel*/
    {
    }

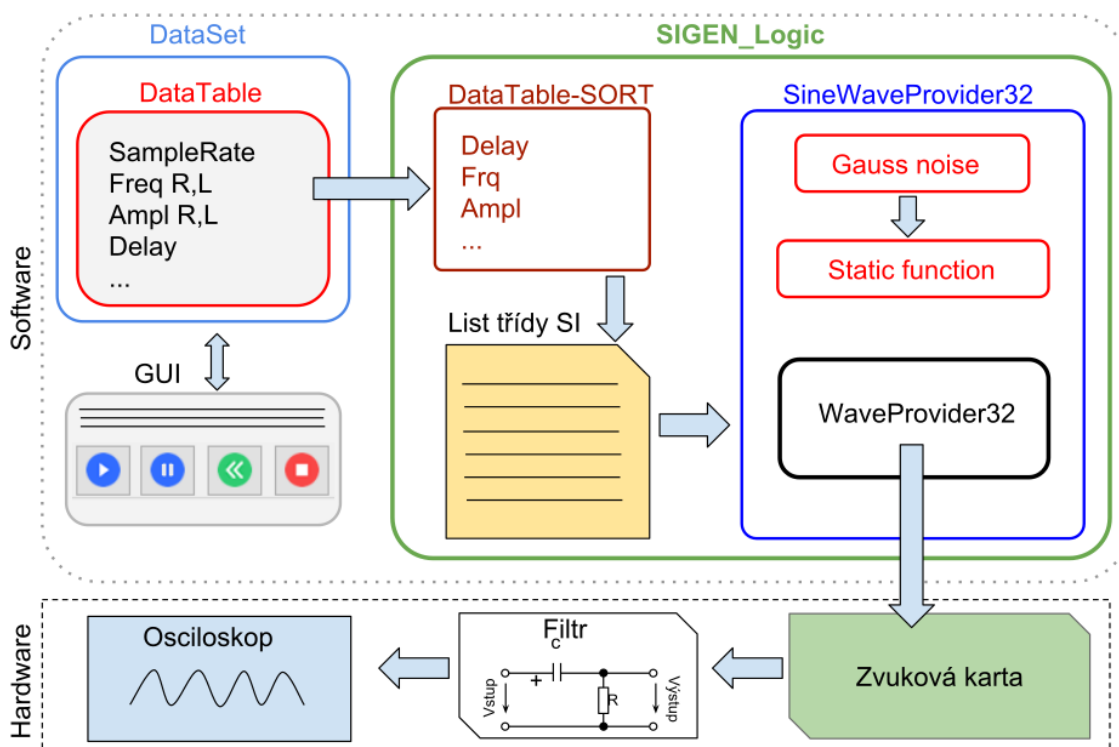
    Počet odkazů: 1 | zbynek fojtik, Před 84 dny | 1 autor, 1 změna
    public WaveProvider32(int sampleRate, int channels)
    {
        SetWaveFormat(sampleRate, channels);
    }
}

```

Obrázek 3.2: Ukázka dědění třídy *IWaveProvider*

IWaveProvider je rodič třídy *SineWaveProvider32*, zdrojový kód této třídy je obsáhlý a proto je k nahlédnutí v příloze této diplomové práce. Tato nová třída zajišťuje matematické výpočty pro průběhy různých signálů. Také je zde naprogramovaná metoda *Read*. S třídou *SineWaveProvider32* pracuje objekt *Sigen_logic*, který zajišťuje obsluhu a ovládání generátoru. Vznik tříd je znázorněn na obrázku 3.3 a 3.4.

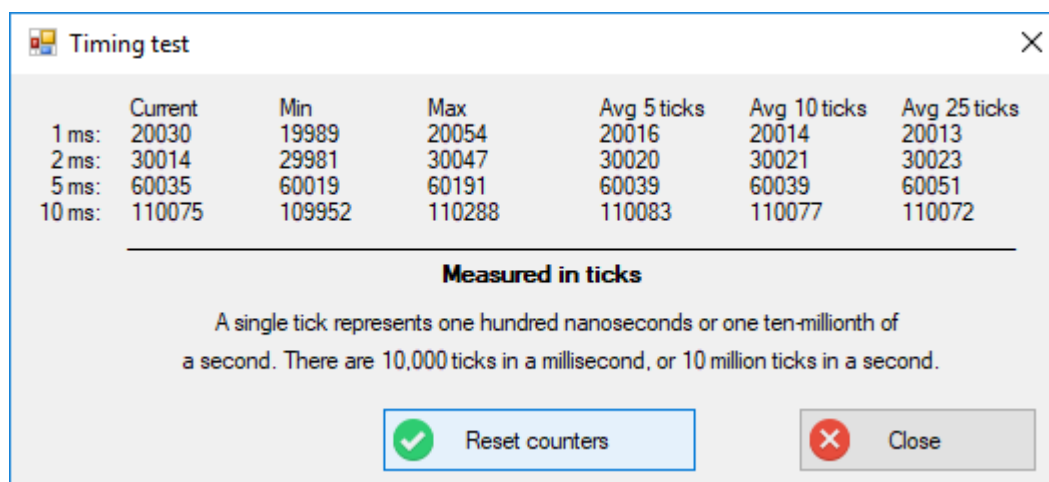
Obrázek 3.3: Schéma dědičnosti a vzniku třídy *SineWaveProvider32* s metodou *Read*



Obrázek 3.4: Schéma realizace generátoru signálu

Jelikož při návrhu hlavního formuláře bylo použito ovládacího prvku `DataGrid`, je potřeba data ukládat do třídy `DataSet`. Z této třídy budou data dále zpracovávána. Třída `DataSet` umožňuje funkci `Sort` a ta nám usnadňuje práci pro spouštění generovaného signálu se zpožděním.

Nyní si můžeme vysvětlit tok dat. Data z `DataSetu` se musí dále setřídít podle hodnoty `delay`, která určuje samotné posunutí signálu v milisekundách. Jakmile jsou data setříděna, stále jsou však pro zpracování robustní a je vhodné zjednodušit jejich zápis, převedeme se do Listu třídy `SignalInfo`. Třída `SignalInfo` obsahuje pouze informace o času spouštění, typu signálu, frekvence, amplitudy. Data z této třídy jsou dále předány do třídy `WaveProvider32` který za použití `NAudio` knihovny data zpracuje a pošle do zvukové karty. Zde jsem narazil na několik nezdarů v podobě špatných matematických vzorců, které bylo nutno opravit, tak i problémy spojené s nesprávným časovým posunem jednoho z generovaných signálů. Problematika časových posunů se ukázala jako velmi zajímavá. A to z hlediska architektury počítače a časové náročnosti na zpracování vlákna. Jelikož v operačním systému běží několik vláken, není možno určit pořadí vykonávaných vláken což je problém přesného časového posunu. Z tohoto důvodu byl do programu přidán `Timing test`. Pomocí tohoto testu je možnost orientačně ověřit přesnost časového zpoždění potřebného pro posun signálu v čase. Tento problém se částečně vyřešil přidáním nového vlákna do programu v podobě `Timeru`, který změní pro potřebnou dobu posunutí, pozastaví příslušný kanál po tuto dobu a následně jej pustí. [6]



Timing test

	Current	Min	Max	Avg 5 ticks	Avg 10 ticks	Avg 25 ticks
1 ms:	20030	19989	20054	20016	20014	20013
2 ms:	30014	29981	30047	30020	30021	30023
5 ms:	60035	60019	60191	60039	60039	60051
10 ms:	110075	109952	110288	110083	110077	110072

Measured in ticks

A single tick represents one hundred nanoseconds or one ten-millionth of a second. There are 10,000 ticks in a millisecond, or 10 million ticks in a second.

Obrázek 3.5: *Timing test - orientační test časového zpoždění*

4 Ověření vlastností realizovaného generátoru

Ověření vlastností realizovaného generátoru jsem prověřil pomocí měření generovaného signálu a také následného porovnání s komerčním produktem. Na začátku jsem si určil měřené rozsahy v mezích parametrů uváděných u zvukových karet. Minimální kmitočet uváděný u zvukových karet je 20Hz a tato hodnota byla také použita jako minimální hodnota generované frekvence. Maximální kmitočet byl zvolen podle maximální uváděné frekvence zvukové karty, 20kHz.

Ověření vlastností jsem rozdělil do tří částí

- Porovnání rozdílu signálu u určité skupiny zvukových karet
- Porovnání průběhu signálů zvukové karty s komerčním generátorem
- Ověření pomocí integračního a derivačního članku

4.1 Porovnání rozdílu signálu určité skupiny zvukových karet

V průběhu testování bylo jasné, že zkoušet rozdíly u tak složitého průběhu jako je hradbový signál, je zbytečné provádět testování až na samotnou maximální uváděnou hranici zvukových karet (20kHz). Hradbový signál jsem porovnával do frekvence 10kHz. Na tomto kmitočtu byly všechny průběhy nevyhovující.

Pro srovnání byl použit tento hardware:

- Desktop Acer Veriton M264(pracovní stanice)
- NTB HP Pavilion dv7 (multimediální)
- NTB Lenovo IdeaPad G770 (multimediální)
- NTB Lenovo ThinkPad P51 (profesionální výkonná stanice)

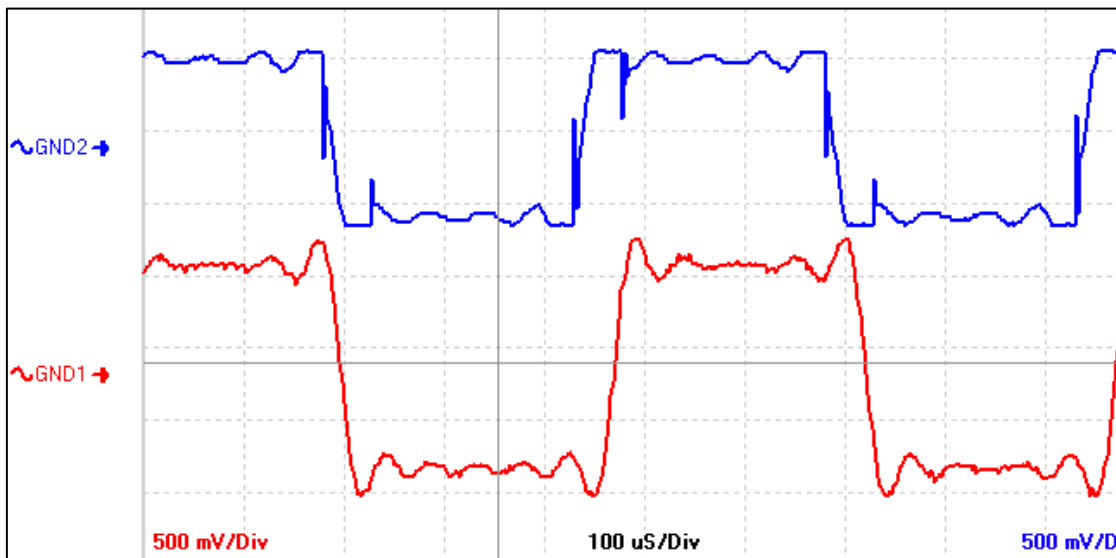
Za použití osciloskopu ELAB-080 jsem měl možnost porovnat vždy pomocí dvou vstupů. Volil jsem tedy hardware do páru a na každém zařízení spustil program SIGEN, na kterém jsem nastavoval stejné parametry pro generování signálu.

- První pár byl sestaven z pracovní stanice ACER a notebooku HP. (Příloha A)
- Jako druhý pár jsem určil dva Lenovo notebooky. (Příloha B)

Z těchto srovnání jsem usoudil, že nejkvalitnější ve srovnání byl Notebook Lenovo IdeaPadG770 a ten jsem následně srovnal s komerčním generátorem.

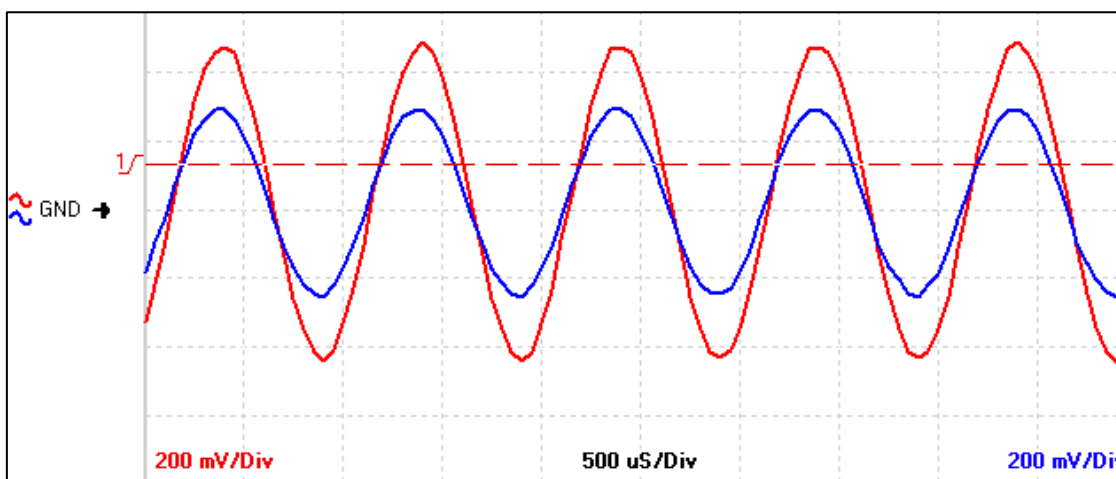
Pro srovnání jsem si určil frekvence od počáteční minimální 20Hz a ž po 10kHz a to konkrétně: **20Hz, 100Hz, 1kHz, 2kHz, 4kHz, 6kHz a 10kHz**. Výsledky jsou uvedeny v příloze, zde příkládám na ukázkou pouze malý vzorek porovnání první dvojice testovaného hardware. Konkrétně se jedná o Desktop PC Acer Veriton M264 a Notebooku HP Pavilion dv7.

Na obrázku grafu je patrné, který hardware produkuje méně kvalitní signál. Celé porovnání je v příloze A. Modrý průběh signálu je generován Desktop PC Acer Veriton M264. Červený průběh signálu pak Notebook HP Pavilion dv7.



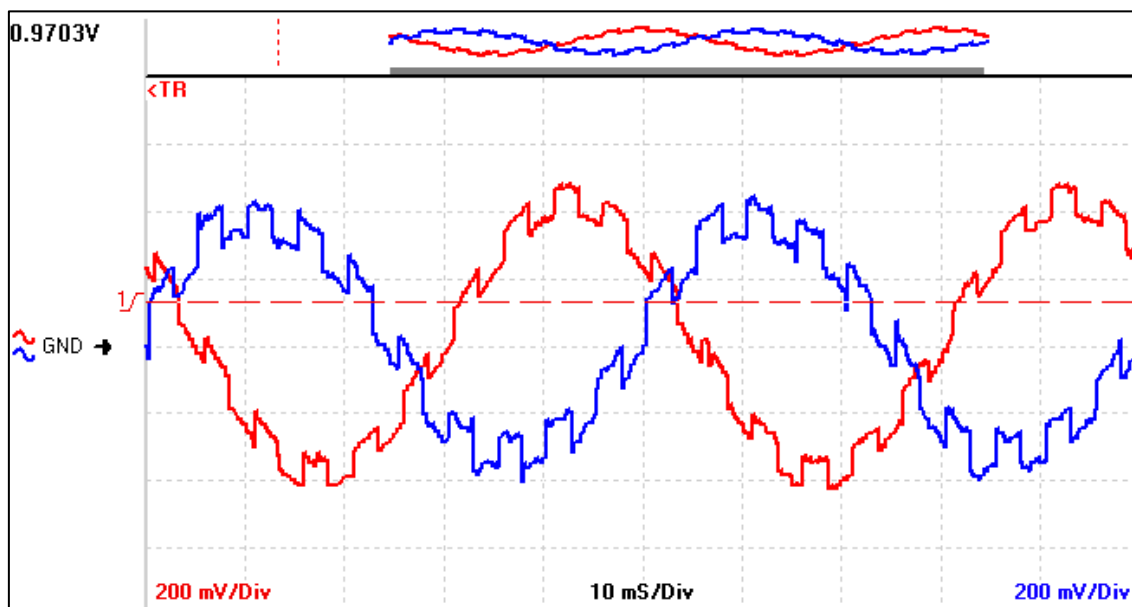
Obrázek 4.1: Znárodnění rozdílů ve hradbovém signálu

Kromě nejnáročnějšího hradbového signálu, jsem porovnával také základní sinusový signál, kde jsem pozoroval amplitudovou závislost některých zvukových karet, na požadované frekvenci. Kdy u 100Hz byla amplituda v kladné polovině cyklu přibližně na hodnotě 400mV, při kmitočtu 1kHz, byla hodnota přibližně 300mV a u 2kHz byla hodnota opět 400mV. Tyto hodnoty vykazovala pouze jedna zvuková karta. Na ukázkou přikládám obrázek. Modrá amplituda je generována notebookem Lenovo ThingPad P51, červená Lenovo IdeaPad G770.



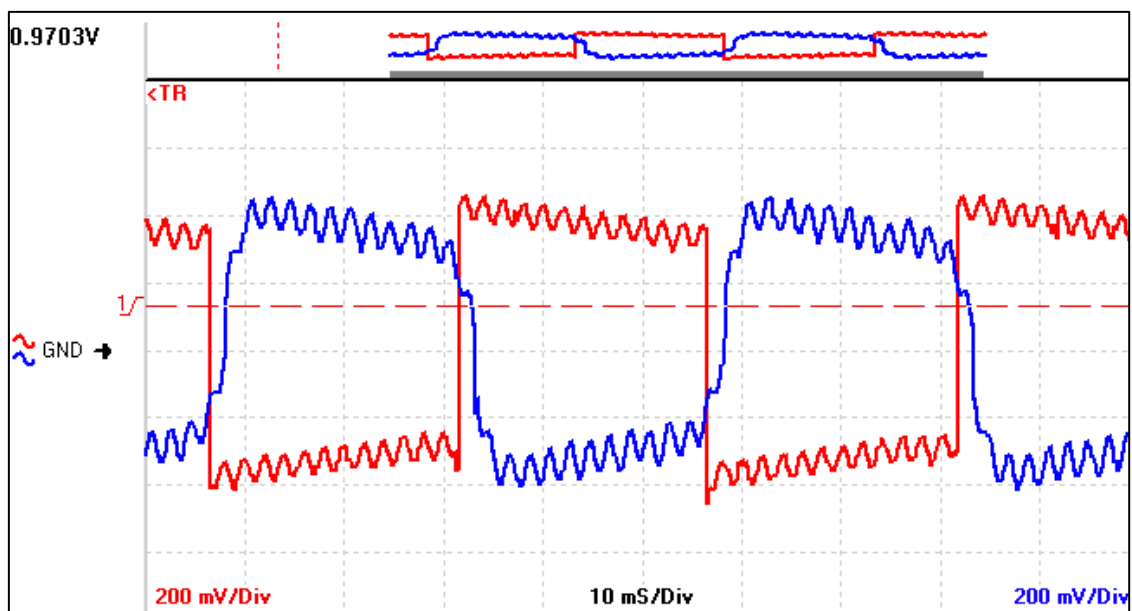
Obrázek 4.2: Změna velikosti amplitudy v závislosti na kmitočtu 1000Hz

Taktéž jsem porovnával i superponování jednoho signálu na druhý, konkrétně na sinusový signál, o kmitočtu 20Hz a amplitudě 1, jsem superponoval hradbový signál o frekvenci 200Hz a amplitudě 0,1. Viz obrázek 4.3.



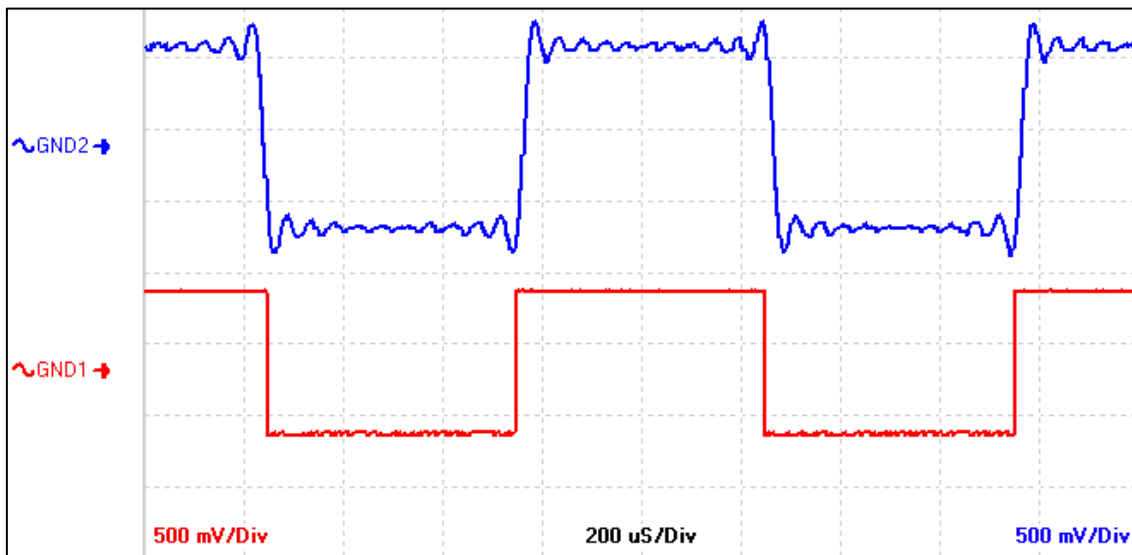
Obrázek 4.3: Superponování hradbového signálu 200Hz na sinus 20Hz

Opačné porovnaní, kdy na hradbový signál při 20Hz amplitudě 1 je superponuji sinus 500Hz, amplitudy 0,1. viz obrázek 4.4.



Obrázek 4.4: Superponování sinus 500Hz na hradbový signál 20Hz

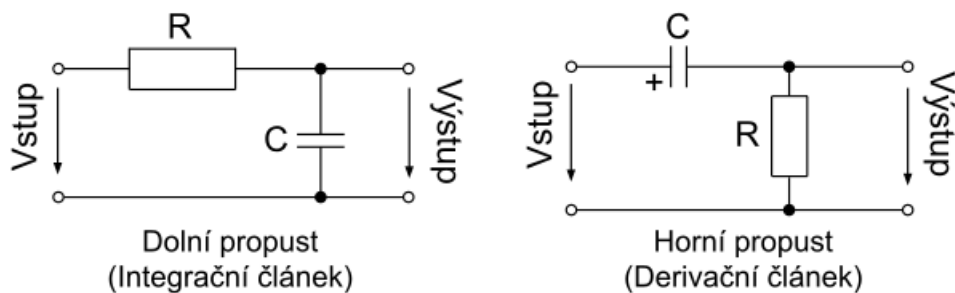
Toto porovnání je velmi jednoduché, přesto zajímavé. Opět jsem zvolil průběh hradbového signálu a porovnával kmitočty **20Hz, 100Hz, 1kHz, 2kHz, 4kHz, 6kHz a 10kHz**. Výsledky jsou uvedeny v příloze, zde příkládám ukázkou porovnání 1kHz na Lenovo IdeaPad G770 v porovnání s komerčním produktem Dino Instruments ELAB-080 , viz obrázek 4.5.



Obrázek 4.5: Srovnání 1kHz proti komerčnímu produktu

4.2 Ověření pomocí integračního a derivačního článku

V poslední části ověření a porovnání funkčnosti generátoru signálu jsem použil klasický integrační článek (dolní propust), a taktéž i derivační článek (horní propust). Dolní propust lépe přenáší nižší kmitočty a vyšší kmitočty potlačuje. Naopak je tomu u horní propusti která lépe propouští vyšší kmitočty a nižší potlačuje. Pro zapojení dolní propusti bylo použito součástek s hodnotou $R=33\text{K}\Omega$, $C=1\mu\text{F}$. Pro zapojení horní propusti byly použity součástky $R=550\Omega$ a $C=1\mu\text{F}$.



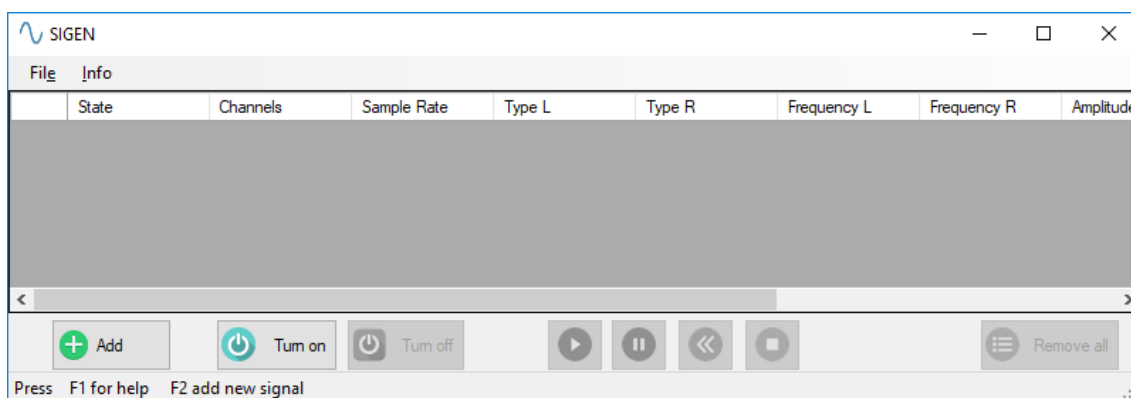
Obrázek 4.6: Schématické znázornění obvodů, dolní a horní propust

4.3 Závěr ověření vlastností navrženého generátoru

Zhodnocení ověření vlastností a funkce realizovaného generátoru vidím jako velmi kladné a uspokojivé. Měření na osciloskopu ukázalo, že hradbové průběhy signálu jakožto nejnáročnější na generování, jsou vyhovující přibližně do 1kHz jak je znázorněno na obrázku 4.5. Sinusový průběh signálu byl vyhovující v celém spektru frekvenčního rozsahu zvukových karet. Zajímavostí bylo, že některé zvukové karty mohou měnit velikost amplitudy podle frekvence, kdy amplituda s rostoucím kmitočtem klesá a o několik frekvencí více, opět začíná stoupat. S tímto poznatkem by se mělo také počítat při používání zvukové karty počítače jako, nízkofrekvenčního generátoru. Porovnání zvukových karet bylo také zajímavou zkušeností. Toto porovnání je dobré pro získání určitého přehledu o kvalitě některých karet.

5 Manuál k programu generování signálu

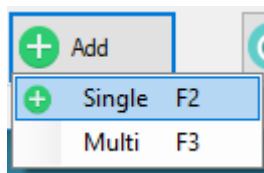
Přesto, že by někomu připadalo ovládání programu intuitivní, jsem se rozhodl, vytvořit přehledný manuál ovládání programu generování signálů. Program jsem pojmenoval SIGEN (Signální Generátor). Zároveň je program lokalizován do anglického jazyka, aby jeho používání bylo snadné pro většinu studentů, jak domácích tak zahraničních. Ještě než začnu popisovat samotný program, cítím potřebu zdůraznit, že k chodu programu SIGEN.exe je zapotřebí knihovna NAudio.dll, která musí být ve stejném adresáři jako samotný program. Ještě bych rád doplnil, že poslední verze programu je možné stáhnout ze stránek programu sigen.kvaline.cz. Po spuštění programu se zobrazí základní okno programu, které můžeme vidět na obrázku.



Obrázek 5.1: Program SIGEN po spuštění

5.1 Vložení signálu

Základním předpokladem po spuštění programu je že bude uživatel chtít vložit parametry signálu pro generátor. Tato funkce se vyvolá stisknutím funkční klávesy F2 pro vložení Single signálu nebo stisknutím funkční klávesy F3 pro vložení Multi signálu. Kliknutím myši na tlačítko Add v levé dolní části programu se rozvine malá nabídka přidání Single nebo Multit.

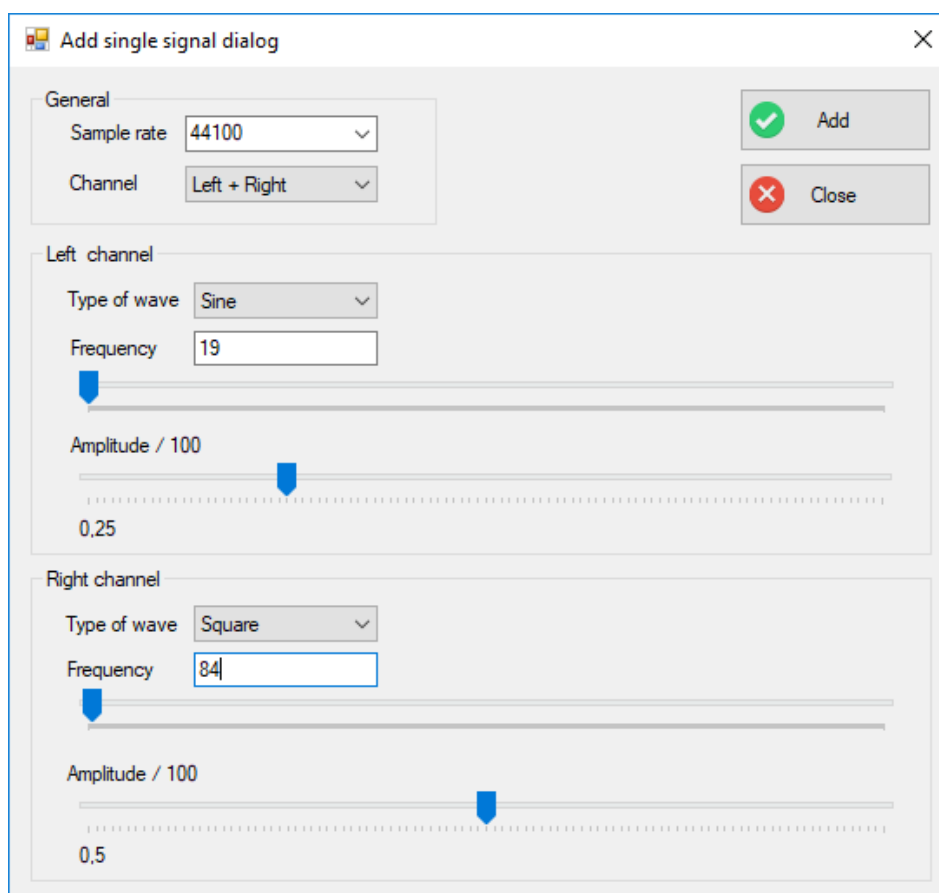


Obrázek 5.2: Zobrazená nabídka typu vložení signálu, Single nebo Multi.

Při výběru možnosti Single se zobrazí zadávací dialog, ten obsahuje všechny potřebné parametry pro generátor signálu. A to konkrétně Sample rate, což je vzorkovací frekvence, po zvolení této vzorkovací frekvence se upraví i maximální hodnota nabízená v posuvníku

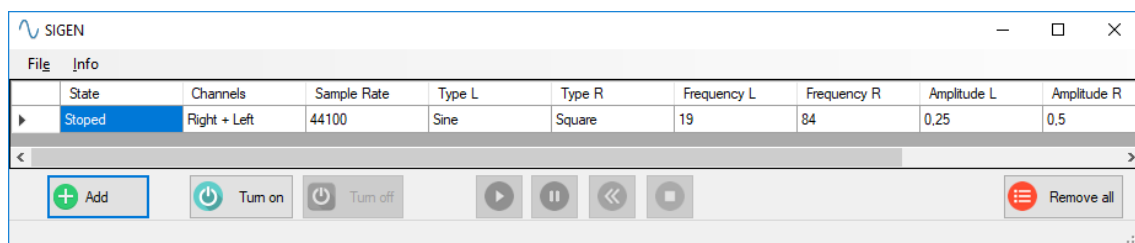
Frequency pro požadovaný kmitočet. Tato frekvence je z důvodu zachování vzorkovacího teoremu poloviční, než je sample rate.

Dále je potřeba nastavit hodnotu Channel což je volba kanálu který bude reprodukovat signál. Volby jsou tři, pravý a levý současně, pouze levý nebo pravý. Jako další krok je nutno z nabídky zvolit typ signálu (sinus, hradbový, šum atd.). V dalším kroku zadat do pole Frequency hodnotu kmitočtu, to můžeme udělat buď, posuvníkem nebo do pole zadat hodnotu ručně. A poslední volbou je velikost amplitudy, tu zvolíme posuvníkem od minimální hodnoty 0.01 až po maximum což je hodnota 1. Add single signal dialog vidíme na obrázku 5.3.



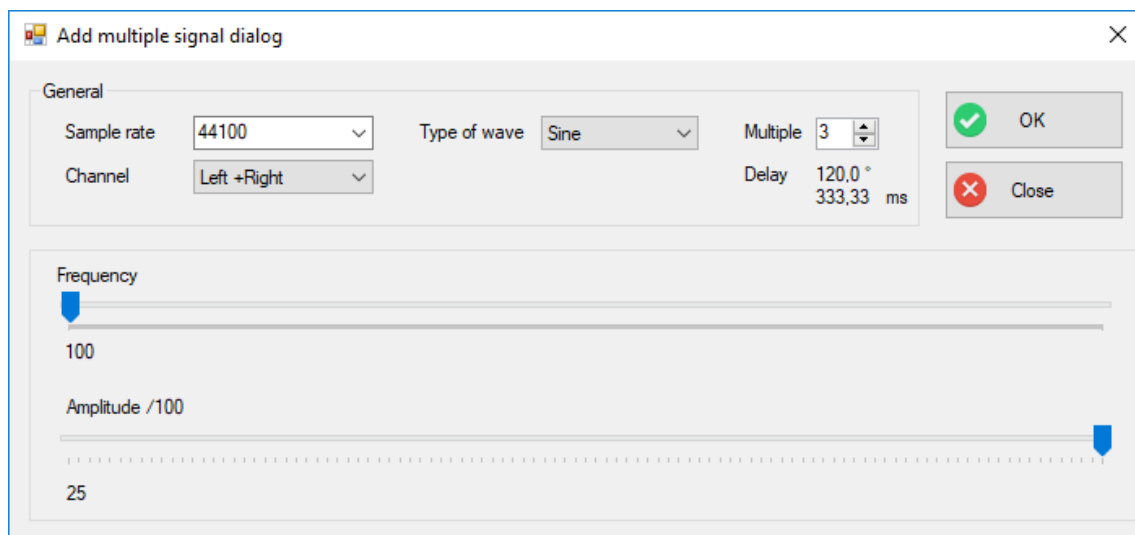
Obrázek 5.3: Zobrazená nabídka typu vložení signálu, Single nebo Multi.

Po potvrzení kliknutím na tlačítko OK se data zobrazí v DataGridu na hlavním formuláři programu SIGEN. Jak vidíme na obrázku 5.4. Zde můžeme numerické hodnoty měnit ručně.



Obrázek 5.4: Zobrazení dat v DataGridu v hlavním okně programu SIGEN

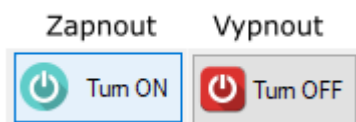
Velmi podobný proces následuje při vložení Multi signálu pomocí klávesy F3, kdy se zobrazí Add multi signal dialog. Zde je pouze malý rozdíl, oproti add simple signal dialogu a to v možnosti určování samostatného signálu pro pravý nebo levý kanál. Sice volba kanálu zůstává zachována, ovšem nikoli nastavení každého zvlášť. Navíc je zde funkce Multiple, která určuje počet signálů ve zvoleném kanálu, počet signálů je od sebe posunut rovnoměrně v 360° , čili pokud zvolíme 3 multi signály sinus, budou vzájemně posunuty o 120° nebo 333,33 milisekund. Viz obrázek 5.5.



Obrázek 5.5: Zobrazení dat v DataGridu v hlavním okně programu SIGEN

5.2 Zapnutí a vypnutí generování signálu

Kliknutím myši na tlačítko Turn ON započne proces generování signálu a odesílání vygenerovaných dat na zvukovou kartu. V průběhu generování signálu, můžeme opět klávesou F2 nebo kliknutím na tlačítko Add přidat další signál, ten se však nepustí automaticky po potvrzení tlačítka OK. Je zapotřebí vypnout generování signálu pomocí tlačítka Turn OFF, v tuto chvíli se generování úplně zastaví. Následným kliknutím na tlačítko Turn ON opět uvedeme generátor signálu k chodu.



Obrázek 5.6: Tlačítka zapnutí a vypnutí generátoru signálu

Na obrázku jsou obě tlačítka aktivní, ovšem v programu je aktivní vždy pouze jedno z nich. Pokud generátor běží, je aktivní tlačítko Turn OFF. Pokud generátor nepracuje, je aktivní tlačítko Turn ON. Tím se nám pustí všechny signály přidané do DataGridu v hlavním okně programu.

5.3 Práce s vybranými signály

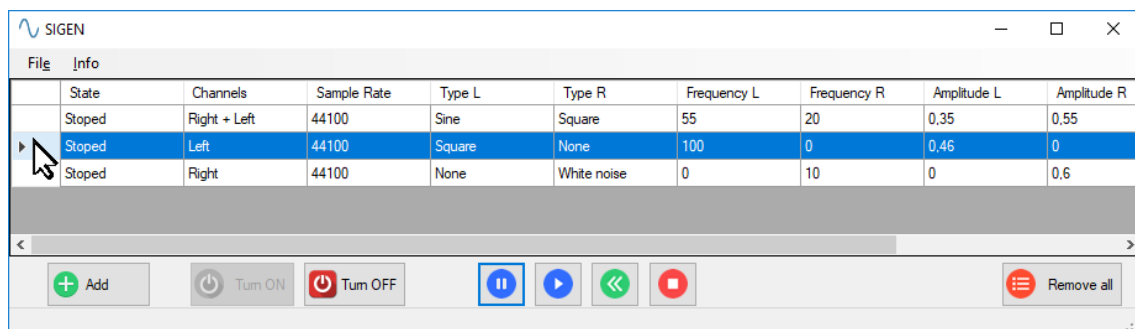
Ve střední části spodního panelu s ovládacími prvky jsou 4 funkční tlačítka, kterými se dá pracovat s vybranými signály. Jsou to základní volby jako je zapauzování signálu, puštění signálu, posun na časové ose a úplné zastavení signálu. Viz obrázek 5.7.



Obrázek 5.7: Funkční tlačítka pro práci s vybranými signály

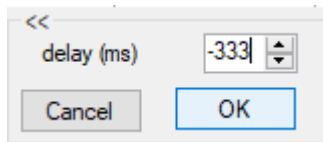
Práce s vybranými signály je velmi jednoduchá, ovšem musí se dodržet pravidlo výběru řádku kliknutím do postranního sloupce v požadovaném řádku, čímž se označí celý řádek, se kterým chceme pracovat. Jako je tomu na obrázku.

Samozřejmě zde v DataGridu funguje i funkce multiselect což znamená, že je možné označit několik řádků najednou a všem hromadně vybraným posouvat čas, pozastavovat nebo je puštět.



Obrázek 5.8: Výběr požadovaného řádku se signálem, který chceme pozastavit, posunout nebo pustit

Funkce pozastavit, pustit a stopnout signál jsou jasné z hlediska podobnosti ovládacích prvku, jak jsme na ně zvyklí u jiných mediálních zařízení. Funkce Delay neboli posunutí v čase je vlastně jakési zpoždění signálu po dobu zvolenou v milisekundách. Při kliknutí na prvek Delay se zobrazí malý dialog s možností zadat čas pomocí šipek nahoru a dolů, ale také může být číselná hodnota napsána na numerické klávesnici.

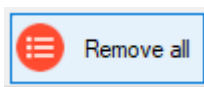


Obrázek 5.9: *Dialog funkce Delay*

Kliknutím na tlačítko OK potvrdíme časové zpoždění daného signálu a tato hodnota se zapíše do tabulky v DataGridu. Stejným postupem můžeme pracovat i s ostatními signály.

5.4 Odebrání všech signálů

V pravé dolní části programu je tlačítko s popiskem Remove all. Toto tlačítko odebere veškerý obsah z tabulky, ve které máme nastaveny všechny parametry k signálům, které mohou nebo jsou generovány.



Obrázek 5.10: *Tlačítko odebrat vše, odebere veškerý obsah DataGridu.*

5.5 Hlavní nabídka

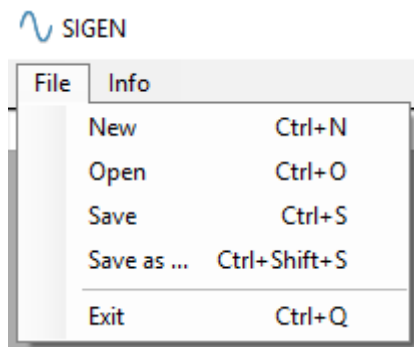
Jen málokterý vývojář nedá do svého programu hlavní nabídku, samozřejmě ji najdeme i zde. Tato nabídka obsahuje dva základní prvky, první je File pro práci se soubory a druhá Info je informativní.

5.6 Nabídka File

V první nabídce najdeme položky pro

- Nový soubor
- Otevření souboru
- Uložení souboru
- Uložení souboru jako
- Konec programu

Funkce všech těchto položek je možno vyvolat klávesovými zkratkami, které jsou uvedeny v pravé části nabídky položek.



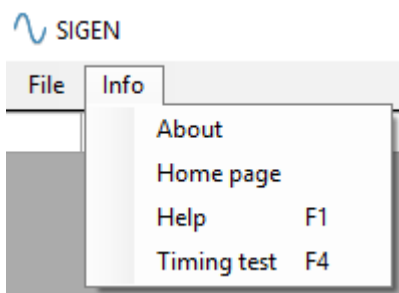
Obrázek 5.11: Hlavní nabídka File a její položky pro práci se soubory

5.7 Nabídka Info

V další nabídce jsou položky čistě informativní, přesto některé informace mohou být pro někoho důležité. Nabídka obsahuje položky

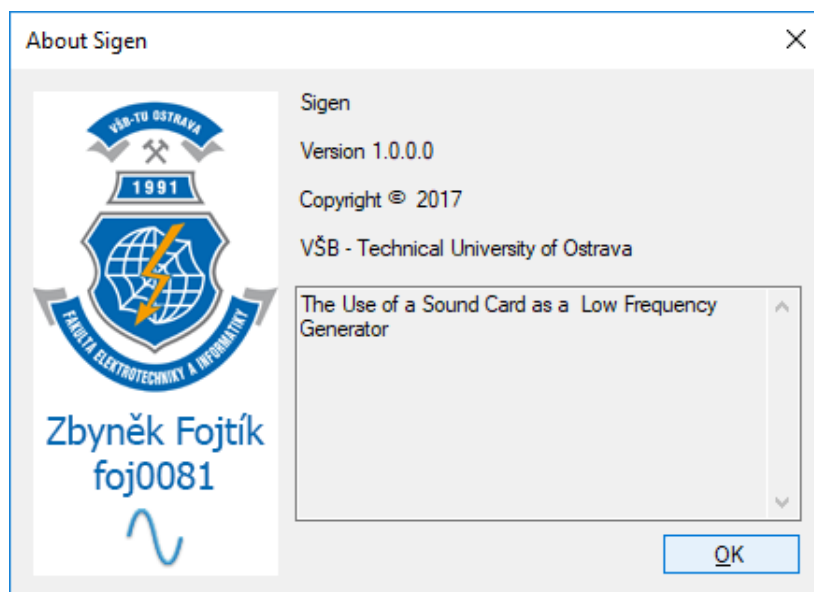
- O programu
- Odkaz na domovskou stránku projektu SIGEN
- Odkaz na soubor s technickou pomocí
- Vyvolání orientačního testu, jenž měří, jak dlouho trvá naměřit požadovaný čas.

Pro úplnost přikládám také obrázek 5.12 této nabídky Info. Zde jsem nepovažoval nutné vkládat klávesovou zkratku ke každé položce.



Obrázek 5.12: Nabídka Info a její informativní položky

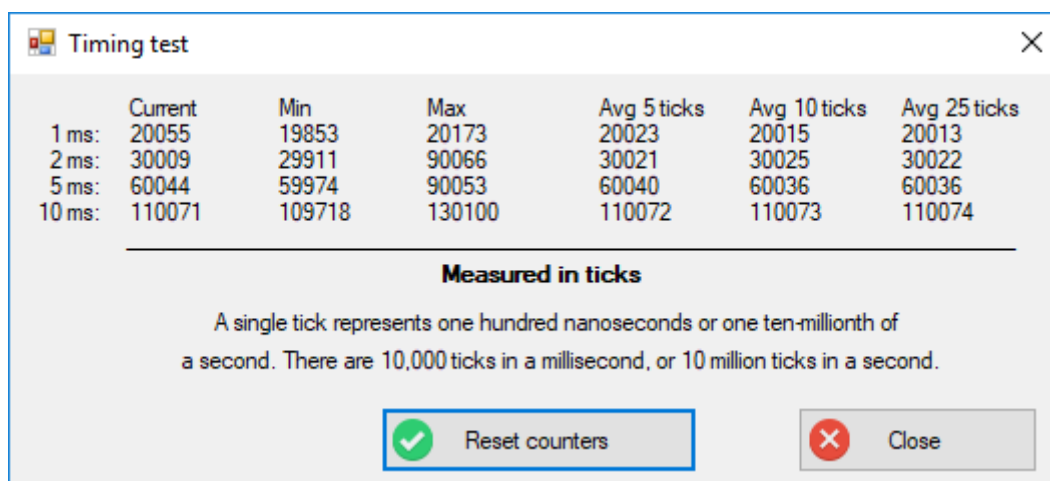
Při kliknutí na položku About se zobrazí klasické okno s informacemi o programu. Název programu, verze, Copyright, rok a další doplňující informace, viz obrázek.



Obrázek 5.13: Okno s informacemi o programu

Kliknutím na položku Home page se spustí internetový prohlížeč a v jeho okně se načte domovská stránky projektu. Na této stránce je odkaz v horním levém rohu na poslední aktualizované vydání programu SIGEN. Položka Help obsahuje stručný manuál k ovládání programu SIGEN. K této položce není sad co více připomínat.

Poslední položkou v nabídce je Timing test, tato položka otevře okno, kde uživatel uvidí jak přesné je počítání času jeho počítače, což může ovlivnit funkci Delay v programu.



Obrázek 5.14: Okno Timing testu s naměřenými hodnotami Ticks.

Samotnou funkci to sice neovlivní, ale nemůžeme chtít, aby program posunul signál v levém kanálu o jednu milisekundu proti pravému, když napočítat jednu milisekundu, pokud někdy trvá zpoždění dvě milisekundy.

Uvedené časy jsou v Ticks. Pro srovnání $10.000 \text{ tiků} = 1 \text{ ms}$. V tabulce jsou vyobrazeny hodnoty zpoždění pro 1, 2, 5 a 10ms v prvním sloupečku. V druhém sloupečku můžeme pozorovat nejmenší naměřenou hodnotu pro daný časový interval. V třetím sloupečku je maximální naměřená hodnota pro daný interval. A v posledních třech sloupcích jsou uváděny průměrné hodnoty uváděny za posledních 5, 10 a 25 tiků. Viz obrázek 5.14.

Závěr

Cílem této diplomové práce bylo navržení a realizování nízkofrekvenčního generátoru, využívajícího zvukové karty počítače. Jednou z částí zadání bylo, aby se dalo přistupovat ke každému kanálu zvlášť a měnit jeho amplitudu, kmitočet a průběh signálu. Program byl navržen a úspěšně naprogramován v jazyce C# v prostředí Visual Studio. Program využívá velmi kvalitní knihovny pro zpracování signálů.

V průběhu vývoje programu jsem si všiml, že ne každý počítač dokáže opravdu přesně napočítat požadovaný čas. Z tohoto důvodu jsem se rozhodl umístit do programu malou testovací funkci, která slouží jako orientační test pro měření jak dlouho trvalo napočítat 1, 2, 5 a 10 milisekund. To pomůže uživateli vytvořit si odhad, o nepřesnosti v posouvání signálu na časové ose. Ovšem navržený generátor signálů, má i možnosti, které nemají ani některé komerční generátory signálu, jako je možnost superponovat signály a to hned několik i více než dva na obou kanálech. A další užitečnou funkcí je možnost ukládat si své projekty a dále se k nim vracet.

Ověření funkčnosti generátoru bylo porovnáno v mezích a možnostech většiny zvukových karet, což je od 20 Hz do 20kHz. A to hned na několika počítačových sestavách. Zde se také ukázalo, že jsou rozdíly ve schopnostech zvukových karet reprodukovat požadovaný průběh signálu. Některé zvukové karty byly dokonce schopny generovat menší nebo i větší kmitočty než uvádí výrobce. Na výsledcích srovnání u více typů zvukových karet je zřejmé, že funkce programu je ovlivněna samotnou kvalitou návrhu obvodů a čipu zvukové karty.

Při srovnání s komerčním generátorem, byli patrné nedostatky v průběhu signálu zvukové karty a to především ve vyšších kmitočtech u hradbového průběhu signálu. Na základní laboratorní úlohy či domácí badatelské pokusy navržený generátor vyhovuje.

Diplomovou práci bych doporučil jako odrazový můstek pro mého nástupce v realizaci nízkofrekvenčního generátoru, pomocí zvukové karty s prostorovým zvukem, jenž využívá 6 kanálů místo dvou jako tomu bylo v mém případě. Díky této diplomové práci získá student přehled o možných problémech a řešeních v této oblasti. Závěrem bych chtěl také říci, že cíl práce byl splněn a celková práce na diplomovém projektu byla velmi zajímavá.

Použitá literatura

- [1] MINASI, Mark. *Velký průvodce hardwarem*. Praha: Grada, 2002. 768s. ISBN 80-247-0273-8.
- [2] BIČOVSKÁ, Blanka. *Elektrická měření*. 1. vyd. VŠB – TUO, 2007, 97s. ISBN 978-80-248-1480-3
- [3] SKAPA, Jan. *Zpracování číslicových signálů*. Pracovní verze. 140s. homel.vsb.cz/ska109/ZCS
- [4] HANÁK, Jan. *C# 3.0 - Programování na platformě .NET 3.5*, 2009, 288s. ISBN: 978-80-7413-046-5
- [5] NEVŘIVA, Pavel. *Analýza signálů a soustav* 1. vyd. BEN, 2002, 354 s ISBN: 80-7300-004-0
- [6] Kolektiv autorů, *Microsoft Visual C# .NET - krok za krokem*. Mobil Media 2002. 654s. ISBN 80-865-932-74
- [7] DOLEČEK, Jartoslav. *Kmitočtové filtry, generátory signálů a převodníky dat*. BEN 2009. 240s. ISBN: 978-80-7300-240-4
- [8] HORÁK, Jaroslav. *Hardware*. Computer Press 2008. 360s. ISBN: 80-251-1741-3
- [9] Electrical Waveforms. [online]. [cit. 2018-4-29]. Dostupné z <https://www.electronics-tutorials.ws/waveforms/waveforms.html>
- [10] NAudio dokumentace [online]. [cit. 2018-4-25]. Dostupné z: <https://github.com/naudio/NAudio/tree/master/Docs>
- [11] D/A převodníky[online]. [cit. 2018-4-25]. Dostupné z: http://noel.feld.cvut.cz/vyu/scs/prezentace2007/DA_prevodniky/

Seznam příloh

Součástí DP je CD/DVD.

Adresářová struktura přiloženého CD/DVD:

Text diplomové práce ve formě PDF, soubor:

Diplomova_prace\DP_Fojtik_foj0081.pdf

Porovnání hradbového signálu na výstupu zvukových karet, 1 skupina:

Priloha \Přiloha A.pdf

Porovnání hradbového signálu na výstupu zvukových karet, 2 skupina:

Priloha \Přiloha B.pdf

Porovnání signálu realizovaného generátoru s komerčním zařízením:

Priloha \Přiloha C.pdf

Porovnání amplitudy v závislosti na frekvenci sinusového signálu:

Priloha \Přiloha D.pdf

FILTR – Dolní propust:

Priloha \Přiloha E.pdf

FILTR – Horní propust:

Priloha \Přiloha F.pdf

Navržený a realizovaný program SIGEN, pro generování signálů:

SIGEN.exe

NAudio.dll

Adresář SaveProjects

V adresáři Source je kopie kompletního projektu v C# v prostředí Visual Studio 2017
